PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-257687
(43)Date of publication of application : 08.10.1993

#1

(51)Int.Cl. G06F 9/38
G06F 9/38
G06F 9/45

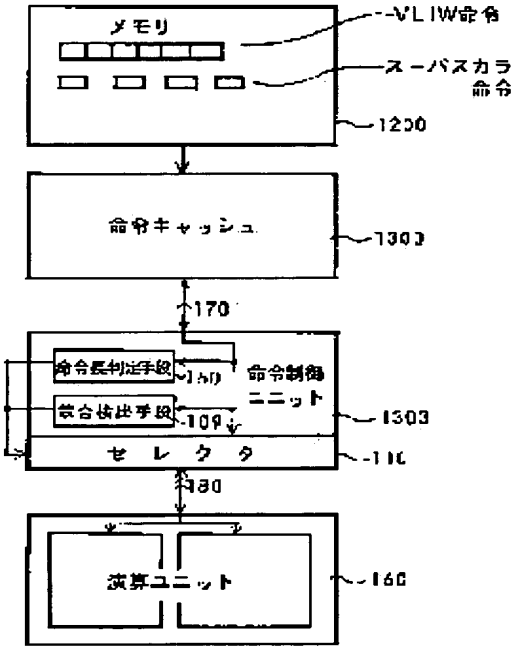
(21)Application number : 04-347929
(22)Date of filing : 28.12.1992

(71)Applicant : HITACHI LTD
(72)Inventor : HOTTA TAKASHI
NAKATSUKA YASUHIRO
TANAKA SHIGEYA
YAMADA HIROMICHI
MAEJIMA HIDEO

(30)Priority
Priority number : 04 29 Priority date : 06.01.1992 Priority country : JP

(54) COMPUTER HAVING PARALLEL ARITHMETIC FUNCTION

(57)Abstract:
PURPOSE: To execute a parallel processing of an operation and to improve the performance by executing plural pieces f instructions of short instruction length for instructing a single operation of one machine cycle, or one piece of instruction of long instruction length for instructing plural operations.
CONSTITUTION: An instruction control unit 1303 reads out an instruction from an instruction cache by using an interface 170, decodes it, and controls an arithmetic unit 160 through an interface 180. The arithmetic unit 160 can process in parallel plural operations. In this case, the unit has a 4-byte length instruction for instructing a single operation, and a 16-byte length instruction for instructing plural operations, and in an instruction cache 1300, the 16-byte length instruction and the 4-byte length instruction are mixed and placed so that there is no competition between the 16-byte length instructions, and between the 16-byte length instruction and the 4-byte length instruction. A competition detecting means 109 detects a competition only between the 4-byte length instructions. In accordance with an output of the competition detecting means 109, a selector 110 selects an operation which can be executed in parallel and decodes it.



LEGAL STATUS

[Date of request for examination] 31.03.1999
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3146707
[Date of registration] 12.01.2001
[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

MicroPatent® Worldwide PatSearch: Record 3 of 3

#1

Family of JP05257687 How It Works

Stage 2 Patent Family - "Extended"		Priorities and Applications		
CC DocNum	KD PubDate	CC AppNum	KD	AppDate
CN 1034604	B 19970416	CN 94101921	A	19940106
		KR 9300074	A	19930106
CN 1094135	A 19941026	CN 94101921	A	19940106
		KR 9300074	A	19930106
DE 69325826	C0 19990909	DE 69325826	A	19930105
		JP 29	A	19920106
DE 69325826	T2 20000224	DE 69325826	A	19930105
		JP 29	A	19920106
EP 551090	A2 19930714	EP 93100067	A	19930105
		JP 29	A	19920106
EP 551090	B1 19990804	EP 93100067	A	19930105
		JP 29	A	19920106
EP 551090	A3 19940907	EP 93100067	A	19930105
		JP 29	A	19920106
EP 855647	A1 19980729	EP 93100067	A3	19930105
		EP 98103921	A	19930105
		JP 29	A	19920106
JP 3146707	B2 20010319	JP 29	A	19920106
		JP 347929	A	19921228
		JP 347929	A	19921228
JP 5257687	A2 19931008	JP 29	A1	19920106
		JP 347929	A	19921228
KR 259738	B1 20000701	JP 29	A	19920106
		KR 9300074	A	19930106
US 5503540	A 19960402	KR 9300074	A	19930106
		US 177972	A	19940106
US 5680637	A 19971021	JP 29	A	19920106
		US 877	B1	19930105
		US 599856	A	19960213
		US 599856	A	19960213

13 Publications found.

Order Selected Documents

US005680637A

United States Patent [19]

Hotta et al.

[11] Patent Number: 5,680,637

[45] Date of Patent: Oct. 21, 1997

[54] **COMPUTER HAVING A PARALLEL OPERATING CAPABILITY**

[75] Inventors: **Takashi Hotta; Yasuhiro Nakatsuka; Shigeya Tanaka; Hiromichi Yamada; Hideo Maejima**, all of Hitachi, Japan

[73] Assignee: **Hitachi, Ltd.**, Tokyo, Japan

[21] Appl. No.: **599,856**

[22] Filed: **Feb. 13, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 877, Jan. 5, 1993, abandoned.

[30] **Foreign Application Priority Data**

Jan. 6, 1992 [JP] Japan 4-000029

[51] **Int. Cl.**[6] **G06F 9/30**

[52] **U.S. Cl.** **395/800**; 395/391; 395/392; 395/384; 395/382

[58] **Field of Search** 395/800, 375, 395/397, 384, 382, 391

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,136,691	8/1992	Baror	395/200
5,163,139	11/1992	Haugh et al.	395/375

FOREIGN PATENT DOCUMENTS

2-130634 of 1990 Japan .

OTHER PUBLICATIONS

Johnson, William *Superscalor Processor Design*, 1991 Prentice Hall, Inc. pp. 5, 6, 25, 105–106, 177, 180–181, 240.

Cohn, R. et al, "Architecture and Compiler Tradeoffs for a Long Instruction Word Microprocessor", 3rd International Conference on Architectural Support for Programming Languages and Operating Systems, 1989, pp. 2–14.

Yoshida et al. (1991) "The Approach to Multiple Instruction Execution in Gmicro/400 Processor" (185–195).

Computer Architecture News, "Software Prefetching", D. Callahan, et al., Apr. 19, 1991, No. 2, New York, New York.

Computer Architecture News, "An Architecture for Software–Controlled Data Prefetching", A.C. Klaiber, et al., May 19, 1991, No. 3, 18th Annual Int. on Computer Architecture, New York, New York.

Proceedings Advanced Computer Technology, Reliable Systems and Applications, "HARP: A VLIW RISC Processor", P.A. Findlay, et al., 5th Annual Computer Conference, Bologna, May 13–16, 1991.

Computer Architecture A Quantitative Approach, "Advanced Pipelining–Taking Advantage of More Instruction–Level Parallelism".

"A Variable Instruction Stream Extension to the VLIW Architecture", A. Wolfe, et al.

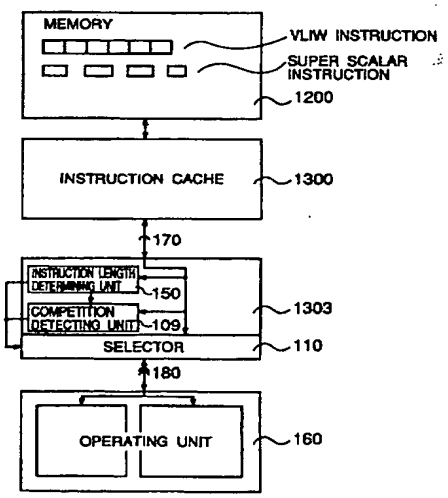
Primary Examiner—Larry D. Donaghue

Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus, LLP.

[57] **ABSTRACT**

A RISC processor is arranged to reduce a code size, make the hardware less complicated, execute a plurality of operations for one machine cycle, and enhance the performance. The processor is capable of executing N instruction each having a short word length for indicating a single operation or an instruction having a long word length for indicating M (N<M) operations. When the number of operations to be executed in parallel is large, the long-word instruction is used. When it is small, the short-word instruction is used. A competition between the long-word instructions is detected by hardware and a competition between the short-word instructions only is detected by software. The simplification of the hardware brings about improvement of a machine cycle, improvement of a code cache hit ratio caused by the reduction of a code size and increase of the number of operations to be executed in parallel for the purpose of enhancing the performance.

7 Claims, 36 Drawing Sheets



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平5－257687

(43)公開日　平成５年(1993)10月８日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/38	３７０　Ｘ	9290－5B		
		３１０　Ｘ	9290－5B		
	9/45				
			9292－5B	Ｇ０６Ｆ　9／44　３２２　Ｆ	

審査請求　未請求　請求項の数17(全 33 頁)

(21)出願番号　特願平4－347929

(22)出願日　平成４年(1992)12月28日

(31)優先権主張番号　特願平4－29

(32)優先日　平４(1992)１月６日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　堀田　多加志
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内

(72)発明者　中塚　康弘
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内

(72)発明者　田中　成弥
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内

(74)代理人　弁理士　小川　勝男

最終頁に続く

(54)【発明の名称】　並列演算機能を有する計算機

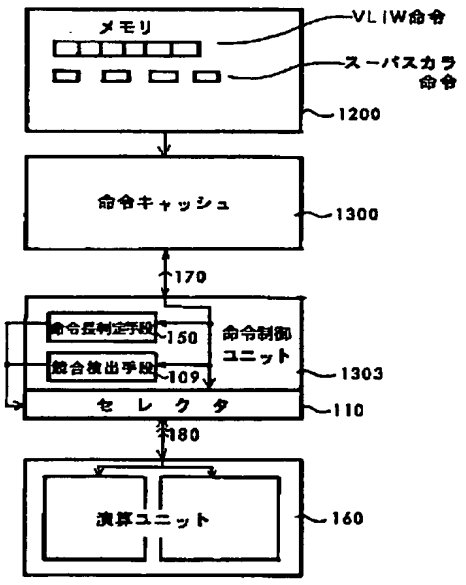
(57)【要約】

【目的】ＲＩＳプロセットにおいて、コードサイズを小さく、またハードウェアを複雑にせず、１マシンサイクルに複数の演算を実行し、性能を高める。

【構成】単一演算を指示する命令長の短い命令をＮ個実行するか、Ｍ個(Ｎ＜Ｍ)の演算を指示する命令長の長い命令を実行することを可能にし、並列に実行できる演算数が大の時には長い命令を、小の時には短い命令を用いる。また、長い命令間の競合はソフトウェアで検出し、短い命令間の競合のみをハードウェアで検出する。

【効果】ハードウェア容易化によるマシンサイクルの向上，コードサイズ縮小によるコードキャッシュヒット率向上、及び並列に実行できる演算数の向上により、性能が高められる。

図　1



1

【特許請求の範囲】

【請求項1】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納されている命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記プログラムカウンタで指示された上記命令が命令長の長い命令であれば上記レジスタに上記命令を設定し、上記プログラムカウンタで指示された上記命令が命令長の短い命令であれば所定のレジスタに上記命令を設定する命令選択手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項2】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると上記レジスタに上記命令を設定し、上記命令長の短い命令と判定され、かつ、上記競合検出手段によって競合がないと判定されると所定のレジスタに上記命令を設定する命令選択手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項3】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記競合検出手段の内容に応じて上記命令長の短い命令を1マシンサイクルに所定の数実行し、上記命令長の長い命令と判定されると、命令長の長い命令を1マシンサイクルに所定の数実行する演算手段とを有することを特徴とする並列演算機能を有する計算機。

2

【請求項4】請求項1，2又は3において、

上記命令は、並列して実行できる演算の多い時には上記命令長の長い命令を、並列して実行できる演算の少ない時には上記命令長の短い命令であることを特徴とする並列演算機能を有する計算機。

【請求項5】請求項1，2，3又は4において、

上記命令長の長い命令と上記命令長の短い命令との使い分けをコンパイラで生成した上記命令を実行することを特徴とする並列演算機能を有する計算機。

【請求項6】請求項1，2又は3において、

上記命令長の長い命令の中で指定できる演算数は、1マシンサイクルで処理される上記命令長の短い命令の数よりも大きいことを特徴とする並列演算機能を有する計算機。

【請求項7】請求項1，2又は3において、

命令長の長い命令と、それ以前の命令長の長い命令との競合が無いようにコンパイラで命令列を生成し、上記コンパイラの出力に基づいて処理を行うことを特徴とする並列演算機能を有する計算機。

【請求項8】請求項1，2又は3において、

有効な命令長の短い命令実行後有効な命令長の長い命令を実行する時、及び、有効な命令長の長い命令の実行後有効な命令長の短い命令を実行する時には、両者の間に一定数の無効命令を挿入することを特徴とする並列演算機能を有する計算機。

【請求項9】請求項1，2又は3において、

命令長の長い命令は、次命令以降に続く無効命令の数を任意に指定できることを特徴とする並列演算機能を有する計算機。

【請求項10】請求項1，2又は3において、

命令長の長い命令の演算結果は、以後の任意の命令から反映することを特徴とする並列演算機能を有する計算機。

【請求項11】請求項10において、

演算結果を反映する一定数後の命令の一定数をN，パイプライン段数をMとした時に、N≧Mとすることを特徴とする並列演算機能を有する計算機。

【請求項12】請求項10において、

過去のレジスタの内容を一定サイクル分保存しておく記憶手段を有することを特徴とする並列演算機能を有する計算機。

【請求項13】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、

3

命令長の長い命令の演算結果は、以後の任意の命令から反映することを特徴とする並列演算機能を有する計算機。

【請求項14】請求項13において、
過去のレジスタの内容を一定サイクル分保存しておく記憶手段を有することを特徴とした並列演算機能を有する計算機。

【請求項15】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、
上記命令には、単一演算を指示する命令長の短い命令と複数演算を指示する命令長の長い命令とを有し、
上記プログラムカウンタで指示された命令が上記命令長の短い命令か命令長の長い命令かを判定する手段と、
上記判定手段の結果が上記命令長の短い命令なら1マシンサイクル中に任意の数の命令を実行し、上記命令長の長い命令なら1マシンサイクル中に1個の命令を実行する手段とを有することを特徴とする並列演算機能を有する計算機。

【請求項16】レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、
上記命令には、単一演算を指示する命令長の短い命令と複数演算を指示する命令長の長い命令とを有し、
上記プログラムカウンタで指示された命令が上記命令長の短い命令か命令長の長い命令かを判定する命令長判定手段と、
上記命令長の短い命令間の競合を検出する競合検出手段と、
上記判定手段の結果が上記命令長の長い命令なら1マシンサイクル中に1個の命令を実行し、上記命令長の短い命令なら上記競合検出手段の結果に応じて、競合が検出されれば、競合が解消されるまで待ってから命令を任意の数実行し、競合が検出されなければ命令を任意の数実行することを特徴とする並列演算機能を有する計算機。

【請求項17】レジスタとメモリとキャッシュメモリとプログラムカウンタを有し、上記プログラムカウンタで示される上記メモリ上の命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、
上記命令は、上記メモリまたは上記キャッシュメモリから上記レジスタへデータ転送を指示する第1のフィールドと、
上記メモリから上記キャッシュメモリへデータ転送を指示する第2のフィールドを有することを特徴とする並列

4

演算機能を有する計算機。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、並列演算を実行する計算機に係り、特にスーパスカラ方式とVLIW方式とを混在させて実行する並列演算機能を有する計算機に関する。

【0002】

【従来の技術】計算機アーキテクチャは、半導体技術の進歩等に支えられ、年々進歩している。1980年代には、これまでの複雑な命令をマイクロ命令を使って複数サイクルにかけて処理するCISC(Complex Instruction Set Computer)に代って、簡単な命令を1サイクルで実行するRISC(Reduced Instruction Set Computer)が現れた。

【0003】さらに、演算方式の高速化技術として、スーパスカラ方式とVLIW (VeryLong Instruction Word) 方式が提案されている。

【0004】スーパスカラ方式とは、命令実行時にハードウェアで命令間の競合を調べ、競合が無ければ1マシンサイクルに複数命令を実行する方式で、特願昭63－283673号(従来技術1)や、J-Hennessy and D.A Patterson "Computer Architecture AQuantiliative Approach" Morgan Kantmann Publishers, Inc1990.P.318（従来技術2）に記載されている。

【0005】またVLIW方式とは、複数演算器の動作を制御するフィールドを持った長い命令を用いる方式である。通常のRISCプロセッサの命令長が32bit なのに対し、64，128，256以上といった長さの命令を持つ。この方式についての説明も、前記J－Hennessy and D.A Patterson(従来技術1）による文献に記されている。

【0006】VLIW方式の改良技術として、1語長命令と3語長命令を混在させて、VLIW方式で処理することにより、コードサイズの大きさを改善する技術が、RobertCohn et al. "Architecture and Compiler Tradeoffs a Long Instruction WordMicroprocessor" Third International Conference on Architectual Supportfor Programming Languages and Operating System, 1989，P．2－14（従来技術3）に記載されている。

【0007】

【発明が解決しようとする課題】以下に、スーパスカラ方式とVLIW方式の特徴について述べる。

【0008】スーパスカラ方式の利点は、単一演算を指示する命令長の短い命令で、有効演算のみを指示するためコードサイズが小さくできる。

【0009】命令を追加する必要がないので前機種との互換性が保たれることである。

【0010】これに対して、スーパスカラ方式の第1の問題点は、並列実行する演算内の競合を検出しなくてはならないことである。並列演算する演算の数が多くなれ

5

ばなる程、競合検出に要するハードウェア量は大となる。

【0011】また、第2の問題点は、現サイクル以前に実行した命令と現サイクルに実行する命令との間の競合検出，待合わせが複雑であることである。並列演算する演算の数が多くなればなる程、現サイクルの命令と競合する可能性のある命令が多くなり、第2の問題点である両者の競合検出，待合わせのハードウェアは複雑になる。

【0012】また、第3の問題点は命令長が短いため、命令によって指定できるレジスタの数が少ないことである。16～32本が典型例である。J－Hennessy and D. A Pattersonの文献のP．325に示されているように、並列して実行可能な演算を増やすためのソフトウェア上の工夫として、ループアンローリングやソフトウェアパイプラインを用いようとすると、レジスタの数が不足する。逆に言えば、存在するレジスタの範囲でしか最適化できない。

【0013】この改善策として、上記従来技術1の文献のE－21～22に、演算結果を次の命令にすぐには反映させないようにすることで、レジスタの数の不足を改善することができると記載されている。

【0014】また、David Callahan et al. "Software Prefetching"Fourth InternationalConference on Architectual Support for Programming Languages and Operating System, 1991, P. 40～52の文献に、スーパスカラマシンにおいて命令によりメインメモリからキャッシュメモリにデータをプリフェッチすることが記載されている。

【0015】以上から、スーパスカラ方式では、命令実行の並列度を増すと上記第1，第2の問題点である競合検出の複雑さからマシンサイクルを高めることができず、処理速度が向上しないという問題を有している。

【0016】次に、VLIW方式の第1の利点は、命令長が長く、1命令の中に複数の演算が指定でき、かつ、命令内での演算間の競合がないため、実行時にハードウェアで、並列実行する演算間の競合を検出しなくてもよいことである。

【0017】第2の利点は、命令長が長いため、多くのレジスタが指定可能なことである。次にVLIW方式の第1の問題点は、前述の命令内での演算間の競合を避けるため、全てのフィールドに有効な演算を指定できるとは限らず、コードサイズが大きくなってしまうことである。

【0018】第2の問題点は現サイクル以前に実行した命令と現サイクルに実行する命令との間の競合検出，待合わせが複雑なことである。これは、スーパスカラ方式の第2の問題点と同じである。

【0019】これについて、ハードウェアでは競合検出を行わず、コンパイラによって予め競合回避を行う技術

6

が、Andrew Wolf and John P. Shen "A Variable InstructionStream Extension to the VLIW Architecture", Fourth International Conference on Architectual Support for Programming Languages and Operating System, 1991, P. 2～14に記載されている。また、VLIW方式の第3の問題点は、前機種との互換性が取れないことである。これは、スーパスカラ方式が、従来の1語長命令をハードウェアで並列実行するのに対し、VLIW方式では、命令の再定義が必要となるからである。

【0020】これまでに述べてきたように、スーパスカラ方式とVLIW方式の利点を活かしながら、スーパスカラ方式とVLIW方式の欠点を補う計算機は存在しなかった。

【0021】本発明の第1の目的は、スーパスカラ方式とVLIW方式を混在させて演算実行可能な計算機を提供することにある。これは、単一演算を指示する命令長の短い命令よりなる従来アーキテクチャを持つ計算機の上位互換性を保ちながら、処理速度を向上させることである。

【0022】

【課題を解決するための手段】上記目的を達成するために、本発明によれば、第1に、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納されている命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記プログラムカウンタで指示された上記命令が命令長の長い命令であれば上記レジスタに上記命令を設定し、上記プログラムカウンタで指示された上記命令が命令長の短い命令であれば所定のレジスタに上記命令を設定する命令選択手段とを有する。

【0023】本発明の第2の特徴によれば、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された上記命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると上記レジスタに上記命令を設定し、上記命令長の

短い命令と判定され、かつ、上記競合検出手段によって競合がないと判定されると所定のレジスタに上記命令を設定する命令選択手段とを有する。

【0024】本発明の第3の特徴によれば、レジスタとメモリとプログラムカウンタを有し、上記プログラムカウンタで指示される上記メモリに格納される命令を読み出し、上記命令の指示する演算を上記レジスタと上記メモリと上記プログラムカウンタに対して実行する並列演算機能を有する計算機において、上記命令は単一演算を指示する命令長の短い命令又は複数演算を指示する命令長の長い命令であって、上記プログラムカウンタで指示された命令が上記命令長の短い命令か上記命令長の長い命令かを判定する命令語長判定手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記命令長の短い命令間の競合を検出する競合検出手段と、上記命令語長判定手段によって上記命令長の短い命令と判定されると、上記競合検出手段の内容に応じて上記命令長の短い命令を1マシンサイクルに所定の数実行し、上記命令長の長い命令と判定されると、命令長の長い命令を1マシンサイクルに所定の数実行する演算手段とを有する。

【0025】

【作用】本発明によれば、1マシンサイクルに単一演算を指示する命令長の短い命令を複数個、あるいは、複数演算を指示する命令長の長い命令を1個実行できるので、演算が並列処理され性能が高められる。

【0026】本発明の一態様によれば、並列して実行できる演算の多い時のみ命令長の長い命令を用いることにより、命令長の長い命令の中の無効フィールドを少なくすることができ、コードサイズを小さくすることができる。これにより、主メモリ及びキャッシュメモリの使用効率が上がり、処理速度の向上が図れる。

【0027】本発明の他の一態様によれば、命令長の長い命令の中で指定する複数の演算内の競合はあり得ず、ハードウェアでこれを検出する必要はない。ハードウェアは同一サイクルに実行する命令長の短い命令間の競合のみを検出すればよい。本発明によれば、1マシンサイクルで実行される命令長の短い命令の数を、命令長の長い命令の中に指示される演算の数より小さくすることにより、平均的に1マシンサイクルに実行される演算数が高い割に、並列に実行する演算間の競合検出を容易にすることができる。

【0028】本発明の他の一態様によれば、命令長の長い命令と、それ以前の命令長の長い命令との競合が無いようにコンパイラで命令列を生成することが可能であり、ハードウェアでこれを検出する必要はない。

【0029】本発明の他の一態様によれば、有効な命令長の短い命令実行後、有効な命令長の長い命令を実行する時、及び、有効な命令長の長い命令の実行後、有効な命令長の短い命令を実行する時には、両者の間に必要な

だけの無効命令を挿入することによりソフト的に両者の競合を解消できるのでハードウェアで両者の競合を検出する必要はない。ハードウェアが検出しなくてはならないのは、現サイクル以前に実行した命令長の短い命令と現サイクルに実行する命令長の短い命令との間の競合だけである。故に、本発明によれば1マシンサイクルで実行される命令長の短い命令の数を命令長の長い命令の中に指示される演算の数より小さくすることにより、平均的に1マシンサイクルに実行される演算数が高い割に、現サイクル以前に実行した、現サイクルに実行した命令との間のハードウェアによる競合検出を容易にできる。

【0030】本発明の他の一態様によれば、命令によって指示された演算結果は直ちに次命令に反映されず、一定数後の命令から反映されるので、命令実行後、その結果が反映されるまでに実行される命令は、書き込まれる前のレジスタの値を読むことができ、ソフトウェアが使うレジスタの数を実質的に多くし、ソフトウェア上の最適化により演算の並列度をあげることができる。

【0031】本発明の他の一態様によれば、競合検出のためのハードウェアが簡単になり、マシンサイクルの向上が図られ、処理速度を高めることができる。

【0032】本発明の他の一態様によれば、単一演算を指示する命令長の短い従来アーキテクチャの命令に、複数演算を指示する命令長の長い命令を追加して新アーキテクチャとできるので新アーキテクチャに従来アーキテクチャを含ませ、上位互換性を保つことができる。

【0033】

【実施例】次に本発明の好ましい一実施例について述べる。発明の本質と無関係な詳細は省略してある。

【0034】図1に全体ブロック図を示す。1200はメモリ、1300は命令キャッシュ、1303は命令制御ユニット、160は演算ユニット、150は命令長判定手段、109は競合検山手段である。命令制御ユニット1303は、インタフェース170を用いて、命令キャッシュより命令を読み出し、デコードし、インタフェース180を通じて演算ユニット160を制御する。演算ユニット160は複数の演算を並列に処理することができる。本計算機は単一演算を指示する4バイト長命令と、複数演算を指示する16バイト長命令を有し、命令キャッシュ1300には、16バイト長命令間、及び、16バイト長命令と4バイト長命令の間の競合は無いように、16バイト長命令と4バイト長命令が混在しておかれている。競合検出手段109は、4バイト長命令間のみの競合を検出する。命令制御ユニット1303は、命令長判定手段150を具備し、16バイト長命令実行時は競合検出手段109の出力を無視し、4バイト長命令実行時には、競合検出手段109の出力に応じて、セレクタ110は並列して実行できる演算を選びデコードし、インタフェース180を通じて演算ユニット160を制御する。尚、ここでは、演算ユニットが2つの場合

を示しているが、2つ以上でも良い。

【0035】以下、レジスタ構成，命令フォーマットを説明し、さらにパイプライン、及び、動作タイミングを説明し、最後に図1の全体ブロック図の詳細を述べる。

【0036】図2にレジスタ構成を示す。FR0～FR31は64ビット長の浮動小数点レジスタ、R0～R31は32ビット長の整数レジスタである。簡単のため、浮動小数点データは全て倍精度で64ビット長とする。また、アドレスは32ビット毎に振られているものとする。

【0037】本実施例では、命令長の短い命令の命令長を1語長，命令の長い命令の命令長を4語長とする。

【0038】図3に、命令形式を示す。1語は32ビットである。基本命令，分岐命令，ロード・ストア命令は1語長命令、複合命令は4語長命令である。基本命令は全てレジスタ・レジスタ間演算である。本実施例では、命令長の長い命令の命令長を4語長としたが、実施例によっては、もっと長いこともあり得る。

【0039】本実施例では、簡単のため4語長命令は必ず4語長境界で区切られた4語に配置されると仮定するが、この仮定をはずすことは容易である。

【0040】まず基本命令について説明する。OPフィールドはオペコードの種類を、S1とS2フィールドは2つのソースレジスタの番号を、Tフィールドはターゲットレジスタの番号を、CCフィールドは、フラグの立て方を示すフィールドである。即ち、S1とS2で示されるレジスタが、OPで示される演算をほどこされ、Tで示されるレジスタに結果が書き込まれる。詳細を図4に示す。

【0041】次に分岐命令について説明する。dはディスプレースメントを示す。分岐命令では、プログラムカウンタPCにdの値が加算される。

【0042】次にロード・ストア命令について説明する。Fフィールドは、ロード、又は、ストアするデータが浮動小数点データであるか、整数データであるかを示す。

【0043】SIZEフィールドは、図4に示すように、ロード、又は、ストアするデータの語長を示す。整数については、1ワードのみが定義され、浮動小数点については、2～16ワードが定義されているものとする。図4に示すように、FST命令では、FR(S1)がR(S2)番地に書き込まれる。SIZEが16ワードの時には、FR(S1)～FR(S1＋7)が、R(S2)番地から始まる連続する16ワードに書き込まれるものとする。また、FLD命令では、R(S1)＋R(S2)番地のデータを、FR(T)に書き込む。SIZEが16ワードの時には、R(S1)＋R(S2)番地から始まる連続する16ワードが、FR(T)～FR(T＋7)に　き込まれる。

【0044】次に、図3，図5を用いて、4語長の複合

命令について説明する。この命令では、I1，I2，IT，SIZE，Fフィールドで示される、ロード・ストア操作と、J1，J2，JT，Jフィールドで示される整数演算と、M1，M2，MTフィールドで示される第1浮動小数点演算と、A1，A2，AT，Aフィールドで示される第2浮動小数点演算と、N1，N2，NTフィールドで示される第3浮動小数点演算と、B1，B2，BT，Bフィールドで示される第4浮動小数点演算と、CC，d，Nフィールドで示されるフロー制御の計7つの演算が指示できる。各フィールドの詳細については、図5に示す。第1浮動小数点演算と、第3浮動小数点演算は乗算、第2浮動小数点演算と第4浮動小数点演算は加減算である。Nフィールドは、本命令に続き、挿入したい無効サイクルの数を示す。使用方法については後で述べる。

【0045】整数演算について説明する。Jフィールド≠1111のときは、図5に示すように通常の演算を行う。しかし、Jフィールド＝1111のときは、データの格納されているメモリからキャッシュメモリへのプリフェッチを行う。すなわち、R(J1)＋R(J2)をアドレスとしてキャッシュメモリをアクセスして、該当するデータが無ければメモリからキャッシュへデータを転送する。

【0046】また、1語長命令では、演算結果は次命令に直ちに反映されるが、4語長命令では、演算結果は3つ後の命令に初めて反映される。この仕様を好ましく利用したパイプライン構成と、プログラム例について以下述べる。

【0047】図6に示すようにパイプライン構成は、IF，D，E，F，Sの5段である。IFステージでは、命令の読み出し、Dステージでは、命令のデコード、Eステージではレジスタの読み出しと演算の一部、Fステージでは、演算、Sステージでは、演算の残りとレジスタへの演算結果の書き込みが行われる。パイプライン構成は、整数演算と浮動小数演算で同じとする。

【0048】図7に1語長命令の本実施例による処理フローを示す。1マシンサイクルに2命令処理されるスーパスカラ方式である。命令1と2，命令3と4，命令5と6，命令7と8のそれぞれが、特に競合の無い限り、並列に処理される。このスーパスカラ方式については、特願昭63－283673号に詳細に記されている。

【0049】次に図8は、命令2の結果を命令3が使う場合の処理の様子を示したものである。命令3と4のEステージは、命令2のSステージが終了するまで引き伸ばされる。前命令の結果が次命令に反映するという命令仕様を満足するため、ハードウェアは上記競合を検出し、図8に示す処理を行わなくてはならない。

【0050】4語長命令の処理の様子を図9に示す。4語長命令は、1マシンサイクルに1命令ずつ処理される。命令1の演算結果は先に述べた仕様により、命令

2，3には反映されず、命令4になって初めて反映される。命令1のレジスタ書き込みステージであるSステージは、命令4のレジスタ読み出しステージであるDステージの1つ前にちょうど終了しているので、図8を用いて説明した様なハードウェアによる競合制御は必要ない。本実施例では、演算ステージはE，F，Sの3段であるが、一般に、演算結果を書き込む前に行う次命令の数Nとパイプライン段数Mの間にN≧M－1であればハードウェアによる競合制御は不要である。本実施例ではN＝2，M＝3のケースである。

【0051】また、1語長命令と次の有効な4語長命令の間には必ず無効な4語長命令を2つおくものとする。同様に有効な4語長命令と次の1語長命令の間には必ず無効な4語長命令を2つおくものとする。

【0052】次にこの4語長命令の好ましいプログラム例について図10，図11，図12を用いて述べる。次式の計算をする場合を考える。

【0053】

$A(i)=A(i)+C\times B(i)$ ，　$1\leq i\leq 24$

但し、Cは定数、A(i)，B(i)は、メモリ上に、図10のように配置されている64ビットの浮動小数点データである。

【0054】図11は、A(i)の計算をするのに、各サイクルに、どんな演算がされるかを説明する図である。横軸は時刻で、単位はマシンサイクルである。図に書かれた横長の箱は、処理されるデータの通る演算器のE，F，Sの3ステージを示す。(1)～(10)について説明する。演算は、インデックスiについて4つずつ行われる。(1)～(10)はA(9)～A(12)を計算するための処理である。以下、各処理について説明する。定数CはFR31にあるものとする。

【0055】(1) A(1)～A(4)をFR4～FR7，B(7)～B(10)をFR0～FR3にロードする。

【0056】(2) FR0×FR31をFR8に格納。

【0057】(3) FR1×FR31をFR9に格納。

【0058】(4) FR4＋FR8をFR12に格納。

【0059】(5) FR5＋FR9をFR13に格納。

【0060】(6) FR2×FR31をFR10に格納。

【0061】(7) FR3×FR31をFR11に格納。

【0062】(8) FR6＋FR10をFR14に格納。

【0063】(9) FR7＋FR11をFR15に格納。

【0064】(10) FR12～FR15をA(9)～A(12)にストアする。

【0065】(1)～(10)の演算スケジューリングに関しては、1つの演算に、図10を用いて説明した様に3サイクルかかることが考慮されている。A(9)～A

(12)の処理について説明したが、A(13)～A(16)，A(17)～A(20)…の処理も全く同様に行われる。各演算器は1サイクルピッチでパイプラインされており、A(13)～A(16)，A(17)～A(20)…の処理をA(9)～A(12)の処理に重ねて、図11のように処理可能である。

【0066】さて、図11に示した処理を実現する4語長命令列を示したのが図12である。A(1)～A(24)の演算は、図12に示す命令1～命令22の22命令で実現できる。使用するレジスタはFR0～FR15，FR31の17本である。命令1，3，5，7，9，11，13，15でどのデータがロードされるか、又、命令12，14，16，18，20，22でどのデータがストアされるかを、図10に示した。命令3はFR0～FR3に値を書き込むか、その結果が反映するのは命令6以降であるので、命令1でロードしたFR0～FR3の値を命令4で使用することが可能である。命令1の結果が、命令2にすぐに反映する従来の方式で図11と同じ処理を行おうとすると、命令3でFR0～FR3に書き込みを行うことはできず、FR16～FR19等、新たなレジスタが必要となる。ところが、使用できるレジスタの数には限りがあり、レジスタの数がネックになり、処理サイクル数が伸びてしまう。図12のプログラムは、現命令の結果が3命令後にしか反映しないという遅延書き込みを生かした結果、17本のレジスタを使うだけで、演算が可能となったのである。

【0067】本例で示した様に遅延書き込みは、レジスタを指定するオペコードのフィールドを増加させずに、実質的に使用できるレジスタの数を増やす効果がある。

【0068】図12上で、“X”印は何も演算するものがない為、空いてしまったフィールドであるが、特に、処理の開始時の命令1～6，終了時の命令17～22に空フィールドが多い。しかし、これらの空フィールドは、コンパイラにより、次の一連の処理の開始処理と、現在の一連の処理の終了処理を重ねることにより、減らすことができる。また、全くすることのない、命令2，4，21は、命令1，3，20のNフィールドを“01”にすることにより、省略することができる。

【0069】本発明では、演算結果を書き込むレジスタの指定は、演算を指定する命令で行うので、命令4，21が初めて省略できる。演算結果を書き込むレジスタの指定を、書き込みが行われるステージに発行される命令で指定する方式では、命令4，21では命令1，18で演算した結果の書き込み指定が必要であり、省略不可である。

【0070】また、図12のプログラムの前に1語長命令があった時に、図12の命令1の前に2つの4語長の無効命令を挿入しなくてはならないが、これは、Nフィールドが“01”である4語長の無効命令を1つ入れればよい。また、図12のプログラムの後に1語長命令が

来る場合、図１２の命令２２のNフィールドを“10”にすればよい。

【００７１】このように命令を長くしてNフィールドを設け、有効に使用することにより、コードサイズを削減することが可能である。また、１語長命令では４語で４つの演算しか指示できないのに比べ、４語長命令では４語で、図５に示すように７つの演算が指示できる。

【００７２】次に、プログラム作成方法について述べる。プログラムは FORTRAN, Cなどの高級言語で記述し、コンパイラにより命令列に変換する。

【００７３】図２６に本発明に関するコンパイラの処理フローを示す。高級言語によるプログラムは、字句解析部，構文解析部，意味解析部を経て、中間コードに変換される。中間コードは、最適化部によって最適化され、コード生成部により図３に示した命令列に変換される。ここで、最適化部とコード生成部を合わせて合成部という。この合成部に本願発明に関する特徴がある。すなわち、この合成部では、中間コードを見て並列してできる演算ができるだけ多くなるように命令列を生成する並列化部を有する。この並列化部では、並列してできる演算数が多いときには４語長命令を用い、並列してできる演算数の少ないときには１語長命令を用いる。ここで、４語長命令を用いるか１語長命令を用いるかの判断基準は、並列して実行できる演算数により決まるが、この演算数はシステムによって異なるので、プログラムを作成するときにパラメータとしてコンパイラに指定できるようになっている。上述のようにすることにより、コードサイズが小さくなり、主メモリ、及び、キャッシュメモリの使用効率が上がり、処理速度が高められる。

【００７４】次に、合成部の特徴としてレジスタの割当てがある。４語長命令はその演算結果が３つ後の命令に初めて反映されるので、特別な配慮が必要になる。例えば、図１２において命令１の結果は命令４に初めて反映されるので、命令２，３では命令１の結果を使用せずにできる演算をできるだけたくさん割り付けておく。この時、演算できる命令が１つも無いときは、無効命令生成部によって無効命令を挿入しておく。また、合成部では、１語長命令と次の有効な４語長命令との間には必ず無効な４語長命令を２つ挿入しておく。逆に、有効な４語長命令と次の１語長命令との間にも必ず無効な４語長命令を２つ挿入しておく。ここで、先に述べたように無効命令はNフィールドを用いることによって省略できる。すなわち、本実施例のコンパイラは、長い命令間の競合を検出し、Nフィールドを用いて、命令実行後に挿入すべき無効サイクルの数を指定できるので、ハードウェアで長い命令間の競合を検出したり、処理をする必要が無い。

【００７５】次に、これまでに説明した命令の処理を行うハードウェアの一実施例について説明する。図１３は図１を詳細化した全体構成である。１３００は命令キャッシュ、１３０１は命令キャッシュコントローラ、１３０２は命令処理フローを制御する分岐ユニット、１３０３は、命令をデコードする命令制御ユニット、1304は整数演算ユニット、１３０７は浮動小数点演算ユニット、１３０６はデータキャッシュ、１３０５はデータキャッシュコントローラ、１３０８はメモリインタフェースユニットである。

【００７６】命令制御ユニット１３０３は、命令キャッシュ１３００より実行すべき命令をパス１３１０を通して受け取り、命令をデコードし、整数演算ユニット制御信号１３１８を整数演算ユニット１３０４に、浮動小数点演算ユニット制御信号1314を浮動小数点演算ユニット１３０７に、分岐ユニット制御信号１３１２を分岐ユニット１３０２に送出する。さらにプログラムカウンタ３５００の制御のためにモード信号１１０も、分岐ユニット１３０２に送出する。また、整数演算ユニット１３０４より、フラグ１３１７を、浮動小数点演算ユニットよりフラグ1315を受け取る。

【００７７】整数演算ユニット１３０４は、オペランドアドレス１３１９をデータキャッシュ１３０６と、データキャッシュコントローラ１３０５に送出する。データキャッシュより読み出されたデータは、データパス１３２０を通して、整数演算ユニット１３０４、又は、浮動小数点演算ユニット１３０７に送出される。データキャッシュの中に所望のデータが無い時には、データキャッシュコントローラ1305が、メモインタフェースユニット１３０８にインタフェース信号１３２１を通して起動をかけ、主メモリよりデータを読み出す。この間の待合わせ制御を、信号１３１６を通して、命令制御ユニット１３０３と行う。

【００７８】分岐ユニットは、次に読み出すべき命令のアドレス１３０９を、命令キャッシュ１３００と、命令キャッシュコントローラ１３０１に送出する。所望の命令が命令キャッシュ１３００の中に無い時には、命令キャッシュコントローラ1301は、メモリインタフェースユニット１３０８にインタフェース信号１３１３を通して起動をかけ、主メモリより、命令を読み出す。この間の待合わせ制御を信号１３１１を通して、命令制御ユニット１３０３と行う。

【００７９】整数演算ユニット１３０４の詳細を示したのが、図１４である。１４００はデコーダ、１４０１は第１ＡＬＵ、１４０２は第２ＡＬＵ、１４０３は整数レジスタである。第１ＡＬＵにはソースパス１４０６，１４０７を通して、整数レジスタファイル１４０３からデータが送られ、演算結果は、ターゲットパス１３２２を通して、整数レジスタファイル１４０３に返される。また、第２ＡＬＵには、ソースパス１４０８，１４０９を通して、整数レジスタファイル１４０３からデータが送られ、演算結果は、ターゲットパス１３１９を通して整数レジスタファイル１４０３に返される。１３１７－１

は、第１ＡＬＵより出力されるフラグ、１３１７－２は第２ＡＬＵより出力されるフラグである。パス１３１９，1322は、ロードストア及びプリフェッチ時のアドレスとしてデータキャッシュ１３０６に送出される。

【００８０】図１３の浮動小数点演算ユニット１３０７の詳細を示したのが、図１５である。１５０１はデコーダ、１５０２は浮動小数点レジスタファイル、１５０３は第１乗算器、１５０４は第２乗算器、１５０５は第１加算器、１５０６は第２加算器である。第１乗算器１５０３には、ソースパス１５１７，１５１８を通して、第２乗算器１５０４には、ソースパス１５１５，１５１６を通して、第１加算器１５０５には、ソースパス１５１３，１５１４を通して、第２加算器１５０６には、ソースパス１５１１，１５１２を通して、浮動小数点レジスタファイル1502よりデータが送られ、演算結果はそれぞれ、ターゲットパス１５０７，１５０８，１５０９，１５１０を通して浮動小数点レジスタファイルに書き込まれる。

【００８１】１３１５－１は、第１乗算器１５０３のフラグ、１３１５－２は第２乗算器１５０４のフラグ、１３１５－３は第１加算器１５０５のフラグ、１３１５－４は第２加算器１５０６のフラグである。

【００８２】図１５の浮動小数点レジスタファイル１５０２の詳細を示したのが、図１６である。１６００～１６０８は浮動小数点レジスタ、１３１４－１～１３１４－９はそれぞれ浮動小数点レジスタ１６００～１６０８の制御信号である。１６１０はロードアライナ、１６０９はストアアライナである。１６１１～１６１８は、浮動小数点レジスタ１６００～１６０８を、メモリと結ぶパスである。パス1611は、ＦＲ０，１６，２４に、パス１６１２は、ＦＲ１，９，１７，２５に接続されている。パス１６１３，１６１４，１６１５，１６１６，１６１７も同様で、パス１６１８にはＦＲ７，１５，２３，３１が接続されている。ロード命令実行時には、パス１３２０を通して、送られてきたデータを、ロードアライナ1610により、１６１１～１６１８の内、所望のパスに乗せかえ、所望のレジスタに書き込む。またストア命令実行時には、１６１１～１６１８にレジスタよりデータが読み出され、ストアアライナ１６０９により、パス１３２０の所望の位置にデータが出力される。

【００８３】図１６の浮動小数点レジスタ１６００の第１の実施例を記したのが、図１７である。レジスタ１６００について示したが、１６０１～１６０８も同様である。図１７に示すようにレジスタ１６００は６４ビットのレジスタの集まりである。１７００～１７６３はそれぞれ１ビットのレジスタである。１５１１－００～１５１８－００はレジスタ１７００の読み出しパス、１５０７－００～１５１０－００はレジスタ１７００の　き込みパス、１６１１－００はレジスタ１７００の読み書きパスである。レジスタ１７６３のパス構成も同様である。

【００８４】図２８は、図１３のデータキャッシュの詳細を図示したものである。２８０１はデータを保持するデータアレイ、２８００はロードストア演算用のアドレスアレイ、２８０２はプリフェッチ用のアドレスアレイである。アドレスアレイ2800と２８０２は同じ内容のデータを保持している。１語長命令のロードストア命令実行時には、パス１３１９または１３２２を用いてアドレスアレイ２８００とデータアレイ２８０１がアクセスされる。４語長命令実行時には、パス１３２２を用いてアドレスアレイ２８００とデータアレイ２８０１がアクセスされる。また、パス１３１９を用いてプリフェッチのためにアドレスアレイ２８０２がアクセスされる。

【００８５】図２９は、ロードストア演算でキャッシュミスを生じたときのパイプラインを示す図である。パイプラインは、データがメモリからキャッシュメモリへ転送される間ロックされる。図２９でφで示されるのがロックされる期間である。

【００８６】一方、プリフェッチ演算のときには、アドレスアレイ２８０２がヒットすれば何も行わない。ミスすれば、そのアドレスを含むブロックがメモリよりデータアレイ２８０１にパスを通して転送される。但し、この時はパイプラインはロックされない。コンパイラにより、ミスする可能性のあるロードストア演算の前にプリフェッチを設定しておけば、メモリからキャッシュメモリへの転送を他の演算と並列に行うことができる。そのために、図２９で示したパイプラインロックによる性能低下を避けることができる。

【００８７】図１７のレジスタ１７００の回路構成例を記したのが、図１８である。1816と１８１７はインバータ、１８０２～１８１５はクロックドインバータである。

【００８８】１３１４－１－１～８がhighになると、それぞれパス１５１１－００～１５１８－００にレジスタの値が出力される。また、１３１４－１－１０～１４がhighになると、パス１５１０－００～１５０７－００の値がレジスタに書き込まれる。また、１３１４－１－９がhighになるとレジスタの値がパス１６１１－００に出力され、１３１４－１－１０がhighになると、パス１６１１－００がレジスタに書き込まれる。信号１８００は予備の読み出しポート、１８０１は予備の書き込みポートである。１８００と１８０１の用途については後に説明する。

【００８９】図１９は、図１６の浮動小数点レジスタ１６００の第２の実施例を示したものである。図１９の実施例は、図１７の実施例と比較して１９００～１９６３の第１シャドウレジスタ，２０００～２０６３の第２シャドウレジスタが付加されている点が異なっている。第１シャドウレジスタ１９００は信号１８００を通して、レジスタ１７００の値を読み取ることができる。また、

信号１９６４を通して第２シャドウレジスタ２０００に、第１シャドウレジスタ１９００の値を送出する。第２シャドウレジスタ２０００は、自分の値を信号１８０１を通してレジスタ１７００に送出する。即、レジスタ１７００～１７６３，第１シャドウレジスタ１９００～１９６３，第２シャドウレジスタ２０００～２０６３はリング状のシフトレジスタを構成している。更に第１シャドウレジスタ１９００～１９６３，第２シャドウレジスタ２０００～２０６３は、レジスタ１７００～１７６３と同様に、バス１６１１－００～１６１１－６３を通して、読み書きができる。

【００９０】１３１４－１－１５は、第１シャドウレジスタ１９００～１９６３の制御信号、１３１４－１－１６は、第２シャドウレジスタ２０００～２０６３の制御信号である。

【００９１】シャドウレジスタの目的は、４語長命令実行時の割込みからの復帰を可能にすることである。図２０～図２２を用いてその動作を説明する。W′ ステージは、レジスタから、第１シャドウレジスタＦＲＳ１に書き込むステージ、W″ は第１シャドウレジスタＦＲＳ１から第２シャドウレジスタＦＲＳ２に書き込むステージである。

【００９２】図２０は割込みのない通常時の４語長命令の動作である。ＦＲ，ＦＲＳ１，ＦＲＳ２のタイムチャート上の数字は、どの命令の演算結果が各レジスタに入っているかを示す。図２０の通り、通常時は、ＦＲからＦＲＳ１へ、ＦＲＳ１からＦＲＳ２へと１サイクルピッチで演算結果がシフトされる。

【００９３】図２１は、命令３と命令４の間に割込みが入った時の動作を示す図である。命令４，５，６，７は無効化される。各レジスタは割込み発生後値の更新を止め、ＦＲは命令３の結果を、ＦＲＳ１は命令２の結果を、ＦＲＳ２は命令１の結果を保持する。また、プログラムカウンタには割込みベクタがセットされる。割込みベクタから始まる割込み処理プログラムで、ＦＲ，ＦＲＳ１，ＦＲＳ２の値をメモリ上に退避する図２２は、割込み処理からの復帰時の動作を説明する図である。まず、割込み処理プログラムの最後に、図２２に示すように命令１の結果をＦＲに、命令２の結果をＦＲＳ２に、命令３の結果をＦＲＳ１に復帰する。こうすることにより、命令４のレジスタ読み出しステージのＥステージで、命令１の結果を見ることができる。命令４のＥステージ終了後、ＦＲの値をＦＲＳ１に、ＦＲＳ１の値をＦＲＳ２に、ＦＲＳ２の値をＦＲへコピーする。この結果、命令５のＥステージでは命令２の結果を見ることができる。命令５のレジスタ読み出しステージのＥステージ終了後も、同じ動作をさせることにより、命令６のレジスタ読み出しステージのＥステージで、命令３の結果を見ることができる。以後の処理は通常通りで、１命令実行毎にＦＲの値をＦＲＳ１へ、ＦＲＳ１の値をＦＲＳ２へコピーし、ＦＲＳ２の値は捨てる。

【００９４】以上、説明した様に、シャドウレジスタを設けることにより、遅延書き込み命令実行時でも割込みを受け付け復帰することができる。シャドウレジスタの無い場合は、図２１で命令３の結果しか退避できず、図２２の割込み復帰時の命令４で、命令１の結果を見ることができない。これは、命令２，３が命令１と同じレジスタに値を書き込む場合があるからである。例えば図１２のプログラムでも、命令１と同じレジスタに命令３で書き込む。

【００９５】次に、シャドウレジスタ付加によるハードウェアの増加量について述べる。レジスタの大きさは、ポート数にほぼ比例する。図１７と図１９を比べれば分かるように、シャドウレジスタのポート数は３と、レジスタのポート数１３に比べてずっと小さいので、シャドウレジスタ付加によるハードウェアの増加量は小さい。

【００９６】図４１は、図１３の命令制御ユニット１３０３の実施例である。１５０は命令語長判定手段、１０１は第１命令レジスタ、１０２は第２命令レジスタ、１０３は第３命令レジスタ、１０４は第４命令レジスタである。４１００はモードレジスタである。１００はモード制御回路、１０５はレジスタ読み出し制御回路、１０６はレジスタ書き込み制御回路、１０７はファンクション制御回路、１０８はパイプライン制御回路、１０９は競合検出回路である。

【００９７】４語長命令は、必ず４語境界をまたがない様に配置されているものとする。また、１語長命令は、２語境界に囲まれた２語が同時に実行されるものとする。本実施例では命令語長判定は、図３で説明した様にオペコードの中の最も左のビット、即、図４１の信号１３１０－１－１(Ｃ０００)そのものを見ることにより行われる。

【００９８】モード制御回路１００の詳細を示したのが図２７である。２７００は制御回路、２７０１はＮフィールド１３１０－４－１を保持するレジスタ、２７０２はディクリメンタ、２７０３はコンパレータである。コンパレータの出力信号VALID（２７０４）は、制御回路２７００に送り出される。レジスタ２７０１にセットされたＮフィールドの値は、ディクリメンタ２７０２により１サイクルごとに１減算され、００になったときに信号ＶＡＬＩＤ(２７０４)がアサートされる。信号ＶＡＬＩＤはネゲート時に無効サイクルの挿入を指示し、アサート時に命令の実行を指示する信号である。

【００９９】制御回路２７００は、競合検出回路出力１１６(ＢＵＢ)と、命令アドレスの下位から２ビット目１３０９－１(ＣＡ３０)と、オペコード中の４語長命令かどうかを示すビット１３１０－１－１(Ｃ０００)と信号２７０４（ＶＡＬＩＤ）を見て５つのモードの内、どれであるかを判定し、図２３に示すように第１～４命令レジスタへのオペコードのセット，プログラムカウンタの

19

インクリメントを信号１１０により行う。また、現サイクルがどのモードであるかを示す信号１１０は、モードレジスタ４１００にラッチされ、その出力信号１３０はレジスタ読み出し制御回路１０５，レジスタ書き込み制御回路１０６，ファンクション制御回路１０７，パイプライン制御回路１０８，競合検出回路１０９に送出される。図２３で、Ｃ０～３は、４語境界内にある４語で、若いアドレスよりＣ０，Ｃ１，Ｃ２，Ｃ３と命令が並んでいるものとする。Ｃ０の最左ビットＣ０００により、図３に示すように、１語長命令か４語長命令かが判定できる。１語命令モード１は、４語境界内の左の２つの命令(Ｃ０，Ｃ１)を実行するモードで、第１命令レジスタにＣ０が、第２命令レジスタにＣ１がセットされプログラムカウンタＰＣは＋２される。また、１語命令モード２は、４語長境界内の右の２つの命令(Ｃ２，Ｃ３)を実行するモードで、第１命令レジスタにＣ２，第２命令レジスタにＣ３がセットされ、プログラムカウンタＰＣは＋２される。即ち、１語長命令実行時には、第１命令レジスタと、第２命令レジスタのみ用い、第３命令レジスタと第４命令レジスタは用いない。４語長命令モードは、４語長命令（Ｃ０，Ｃ１，Ｃ２，Ｃ３）実行するモードで、第１～４命令レジスタにＣ０～Ｃ３がセットされ、プログラムカンタＰＣは＋４される。競合モードとは、競合検出回路１０９が競合を検出した場合で、第１～４命令レジスタ及び、モードレジスタ４１００は前サイクルの値を保持する。また、プログラムカウンタＰＣの更新は行わない。無効命令モードは、現サイクル以前に実行した４語長命令のＮフィールドで、現サイクルにハードウェアで無効命令(ＮＯＰ)を挿入することを指示されている場合で、命令レジスタには無効命令がセットされ、プログラムカウンタＰＣは更新されない。これにより、無効サイクルが１サイクル挿入されることになる。

【０１００】１語長命令実行時にはＣ０、又は、Ｃ２の実行の為に、第１ALU1401(図１４)，第１乗算器１５０３(図１５)，第１加算器１５０５(図１５)を用いる。一方、Ｃ１、又は、Ｃ３の実行の為に、第２ALU1402(図１４)，第２乗算器１５０４(図１５)，第２加算器１５０６(図１５)を用いる。また、４語長命令実行時には、ロードストア演算のアドレス計算を第１ALU1401（図１４）で、整数演算を第２ALU1402（図１４）で、第１浮動小数点演算を第１乗算器１５０３(図１５)で、第２浮動小数点演算を第１加算器１５０５(図１５)で、第３浮動小数点演算を第２乗算器１５０４(図１５)で、第４浮動小数点演算を第２加算器１５０６(図１５)で行う。

【０１０１】図４１のレジスタ読み出し制御回路１０５，レジスタ書き込み回路１０６，ファンクション制御回路１０７は、モード制御回路出力のモード指定信号１１０と第１～４命令レジスタの値により、上述の演算器割当て規則に従い、整数演算ユニット１３０４(図１３)

20

の制御信号１３１８を生成する。レジスタ読み出し制御回路について、更に詳細に説明したのが図２４である。６つの演算器のそれぞれの２つの入力に入れるレジスタの指定を、オペコードのどのフィールドで行うかを示している。フィールドの略号については、図３の複合命令の欄に示す。４語長命令のＪ１とＡ１は、１語長命令のＳ２と、４語長命令のＪ２とＡ２は、１語長命令のＳ２と同じ位置であることを利用し、図２４上のフィールド指定には、１語長命令時も、Ｊ１，Ｊ２，Ａ１，Ａ２を用いて述べている。これは、Ｃ０とＣ１の区別の為である。

【０１０２】次に図４１の競合検出回路１０９について述べる。図７～図９を用いて説明したように、本実施例では４語長命令間の競合を検出する必要はない。また、本実施例では、１語長命令実行用の全ての演算器を２重化している為、同時に実行する２つの１語長命令間の演算器による競合はあり得ない。簡単のためにレジスタ競合も無いものとする。本実施例をレジスタ競合がある場合に拡張することは、例えば、特願昭63－283673号のように容易である。前に述べたように、１語長命令と次の有効な４語長命令の間に、必ず無効な４語長命令が２つ入り、同様に、有効な４語長命令と次の１語長命令との間には必ず無効な４語長命令が２つ入っているので、４語長命令と１語長命令間の競合も有り得ない。故に、競合検出回路１０９は、現サイクルの１語長命令と、それ以前に実行された１語長命令間の競合のみを検出すればよい。モード制御回路１００の制御により、１語長命令は、第１命令レジスタ１０１と第２命令レジスタ１０２にのみセットされるので、競合検出回路１０９は、第１命令レジスタ１０１と第２命令レジスタ１０２のみを見ればよく、第３命令レジスタ１０３，第４命令レジスタ１０４は見る必要はない。

【０１０３】図２５は、競合検出回路１０９の実施例のブロック図である。２５０１～2504はレジスタ、２５０５はマスク回路、２５０６～２５２１はコンパレータである。図７において、命令７，８を現命令とし、命令３～６との競合検出を考える。命令１，２のＳステージの次に命令７，８のＥステージが来るので、命令１，２と命令７，８の間の競合はない。図２５のレジスタ２５０１には、命令５が書き込むレジスタの番号が、レジスタ２５０３には、命令６が書き込むレジスタの番号が、レジスタ２５０２には、命令３が書き込むレジスタの番号が、レジスタ２５０４には命令４が書き込むレジスタの番号が記憶されている。上記４つのレジスタと、命令７及び命令８が読み出す４つのレジスタの番号を、２５０６～２５２１の１６個のコンパレータで比較し、結果をマスク回路２５０５に送出する。マスク回路２５０５は、モード制御回路１００の出力１３０や、パイプライン制御回路１０８の出力１１５を見て、コンパレータのヒット信号が有効であるかどうかを判定し、有効であれ

ば競合を示す信号１１６をアサートする。即、コンパレータの出力がレジスタの一致を示していてもその命令が無効化される場合は、信号１１６をネゲートする。このマスク回路２５０５により、モード信号１３０が４語長モードであることを示している時には、信号１１６をネゲートする。

【０１０４】次にパイプライン制御回路１０８について説明する。パイプライン制御回路は、モード信号１３０，図１３の整数演算ユニット１３０４からのフラグ信号1317，図１３の浮動小数点演算ユニット１３０７からのフラグ信号１３１５，図１３のデータキャッシュコントローラとのインタフェース１３１６，図１３の命令キャッシュコントローラとのインタフェース１３１１を用いて、図１３の分岐ユニット１３０２制御信号１３１２を送出し、分岐ユニットを制御する。即ち、有効な分岐命令が来た時には、分岐を行い、それ以外の時には、モード信号１１０を用いて、図２３のように分岐ユニット内にあるプログラムカウンタを制御する。またパイプライン制御回路１０８は、信号１１５を、レジスタ読み出し制御回路１０５，パイプライン書き込み制御回路１０６，ファンクション制御回路１０７，競合検出回路１０９を送出し、パイプラインの状態を制御する。即ち、命令キャッシュ、あるいはデータキャッシュのアクセスに際してミスを生じた時に図２９に示すようにパイプラインをロックする。

【０１０５】次に上記実施例の第１の変形例について述べる。上記実施例において、４語長命令は演算結果を３つ後の命令に初めて反映することにより、４語長命令間の競合検出部が不要になった。同様の効果を達成するために、４語長命令でも演算結果を次の命令に反映するが、４語長命令間での競合をコンパイラで避けるようにすることもできる。具体的には、ある４語長命令が書き込むレジスタを次の４語長命令と次の次の４語長命令とで読まないようにするのである。このようにすることによって、第１の実施例のレジスタ数を実質的に増やす効果は失われるが、図１９で示したシャドウレジスタは不要になる。

【０１０６】さらに、上記実施例の第２の変形例について述べる。図３で示した実施例では、命令の中に１語長命令か４語長命令かを示すビットを持っていたが、計算機の中に１語長命令か４語超命令かを示すフラグを持ち、このフラグを命令で制御することも可能である。これによって、フラグを制御する命令が必要になるが、１度フラグを切り換えれば毎回命令の中で語長を示す必要が無くなるという利点がある。

【０１０７】図３０～図３２を用いて、上記実施例の第３の変形例について述べる。

【０１０８】本変形例では、図３０の様に浮動小数点レジスタを３２本から１２８本に拡張している。図３１に命令フォーマット，図３２に命令の説明を示す。ＦＲ０～３１は基本命令，複合命令の両者で使用可であるが、ＦＲ３２～ＦＲ１２７は、複合命令でのみ使用するレジスタである。レジスタを指定するＩ１，ＩＴ，ＭＴ，Ａ１，ＡＴ，Ｎ１，ＮＴ，Ｂ１，ＢＴの各フィールドは図３１に示すように、各７ビットと増加する。本変形例では、ソースレジスタの片方をターゲットレジスタと一致させることにより、全体を４語に収めているが、そうしたくなければ、全体の語長をさらに長くすることも可能である。

【０１０９】本変形例では、基本命令に複合命令を追加し、複合命令で基本命令が使えるレジスタの数よりも、多くのレジスタを使えるようにすることにより、全体として、使用可能なレジスタ数を増やせるという効果がある。本変形例ではＦＲ０～３１は基本命令と複合命令の両方でアクセス可であるが代案として、基本命令用の３２本と、複合命令用の１２８本のレジスタを独立のものとすることも可能である。

【０１１０】さらに本変形例では、図３２に示すように、メモリからキャッシュへのプリフェッチの際にプリフェッチする語数をＪＴフィールドで指定可となっている。これにより一度に複数のブロック転送を行うよう命令で指示でき、効率があがるという利点がある。

【０１１１】図３３，図３４を用いて上記実施例の第４の変形例について述べる。本変形例では、データのプリフェッチをＪ１，Ｊ２，ＪＴといった整数演算フィールドで行わずに、１ビットのＰフィールドで行っている点が異なる。Ｐ＝１の時には、ロード・ストア演算に用いられたアドレスを含むブロックの次のブロックがプリフェッチされる。こうすることにより、プリフェッチに要するフィールドが節約でき、ロード・ストア演算，整数演算，プリフェッチの３動作が並列指定可となるという利点が生まれる。

【０１１２】図３５はもう１つの実施例について説明する全体ブロック図である。

【０１１３】３５００はプログラムカウンタ、３５０１は命令を格納するメモリ手段、3502はマスク・スイッチ回路、３５０３～３５０６はM個のｎバイト長の命令レジスタ、３５０７はデコーダ、３５０８，３５０９はL個（Ｌ≧１）の演算ユニット、１５０は命令長判定手段、１０９は競合検出回路、１００はモード制御回路、４１００はモードレジスタである。

【０１１４】プログラムカウンタ３５００は命令アドレス３５１３を、命令を格納するメモリ手段３５０１に送出する。３５０１の中には、ｎバイト長の命令とｎ×Mバイト長（Ｍ＞１）の命令が混在しており、命令アドレス３５１３で指定された命令を含む複数の命令をマスク・スイッチ回路３５０２に送出する。マスク・スイッチ回路３５０２はｎバイト命令であればM個の命令レジスタの内N個（１≦Ｎ＜Ｍ）の命令レジスタ３５０３～３５０４の内の少なくとも１つにセットし、ｎ×Mバイト

令命であれば命令レジスタ３５０３～３５０６にセットする。デコーダ３５０７は命令レジスタ３５０３～３５０６よりの命令３５１９～３５２２をデコードし、Ｌ個の演算ユニットを制御信号３５２３，３５２４を用いて制御する。命令長判定手段１５０は、少なくとも命令３５１４の一部を見て、命令長を示す信号３５２６をモード制御回路１００に送出する。競合検出回路１０９は、命令レジスタ３５０３～３５０４を見て、ｎバイト命令間の競合の有無を知らせる信号１１６をモード制御回路１００に送出する。モード制御回路は、命令長，競合の有無，プログラムカウンタの値によりモードを判定し、制御信号１１０により、プログラムカウンタ，マスク・スイッチ回路，デコーダを制御する。

【０１１５】本実施例と、図１～図２９及び図４１の実施例との対応を説明する。図１～図２９の実施例は、ｎ＝４，Ｍ＝４，Ｎ＝２，Ｌ＝２の場合である。また演算ユニットは整数演算ユニットと浮動小数点演算ユニットであった。また、図３５のマスク・スイッチ回路は、図４１の第１～４命令レジスタ１０１～１０４にセットする命令を生成しているセレクタや、ＮＯＰによるマスク等に対応する。また図３５の競合検出回路１０９は、図４１の競合検出回路１０９に対応する。図３５の命令長判定手段１５０は図４１の命令長判定手段１５０に対応する。図３５のモード制御回路１００は、図４１のモード制御回路１００に対応する。

【０１１６】図３，図５，図１０，図１１，図１２で示した実施例では、図１０の様にメモリ上でのデータ配置が制約されるという欠点があった。これを解決したのが図３６～図４０に示す実施例である。本実施例の複合命令では、図３６，図３７に示すように、１命令でロード・ストア等のメモリ演算を２個実行できる。ハードウェアとしてはキャッシュを２ポート化するか、あるいは、１マシンサイクルに２度アクセス可能な構成にすればよい。図３９，図４０に図１１，図１２に示したものと同じ問題を解くプログラムを示す。

【０１１７】これまでに説明した実施例では、命令長の短い命令と長い命令を混在した計算機において、長い命令の演算結果を、以後の任意の命令から反映する。あるいは、長い命令は、次命令以降に続く無効命令の数を任意に指定できる。あるいは、長い命令は、メモリまたはキャッシュメモリからレジスタへデータを転送する第１のフィールドと、メモリからキャッシュメモリへデータを転送する第２のフィールドを設けるといった工夫を行ったが、これらの工夫は、長い命令のみを有するＶＬＩＷ型計算機に対しても有効である。

【０１１８】

【発明の効果】以上説明したように、本実施例では、４語長命令を用いて、４語で７つの演算を指定できるが、競合検出は、４×４＝１６個のコンパレータで行うことができる。４語長命令間の競合検出をハードウェアで行

おうとすると、前サイクルの分岐演算を除く、６個の演算の書き込みレジスタと前々サイクルの同じく６個の書き込みレジスタと、現サイクルの１２個の読み出しレジスタとの間の競合検出が必要で、（６＋６）×１２＝１４４個のコンパレータが必要となる。本実施例では、これに比して１６／１４４のハードウェアで済むという利点がある。

【０１１９】本実施例では、４語長命令で指示される演算数７に対して、１マシンサイクルで処理される命令長の短い命令が２である。これにより、２演算分の競合検出回路で最大７演算の並列処理が得られるという効果がある。

【０１２０】本発明によれば、並列に実行する演算の数を増やし、性能を高めることができる。

【０１２１】本発明によれば、コードサイズを小さくすることができる。これによりコードキャッシュのヒット率が高まり、性能を高めることができる。

【０１２２】本発明によれば、並列に実行する演算内のハードウェアによる競合検出を容易にすることができる。これにより、マシンサイクルを高めること、ハード物量を減らし、コストを下げることができる。特に、長い命令の中で指定する演算数が大の時、この効果は著しい。

【０１２３】本発明によれば、現サイクル以前に実行した命令と、現サイクルに実行する命令との間のハードウェアによる競合検出；待合わせを容易にすることができる。これにより、マシンサイクルを高めること、ハード物量を減らし、コストを下げることができる。

【０１２４】本発明によれば、ソフトウェアが用いられるレジスタの数を実質的に多くし、ソフトウェア上の最適化により演算の並列度をあげ、性能を高めることができる。

【０１２５】本発明によれば、従来アーキテクチャとの上位互換性を保つことができる。

【図面の簡単な説明】

【図１】命令制御ユニットの全体図である。

【図２】レジスタ構成を示す図である。

【図３】命令形式を説明する図である。

【図４】１語長命令の動作を説明する図である。

【図５】４語長命令の動作を説明する図である。

【図６】パイプラインステージを説明する図である。

【図７】競合無の時の１語長命令処理のパイプラインを示す図である。

【図８】競合有の時の１語長命令処理のパイプラインを示す図である。

【図９】４語長命令のパイプラインを示す図である。

【図１０】データのメモリ上での配置を示す図である。

【図１１】４語長命令を用いた時の演算の様子を説明する図である。

【図１２】４語長命令を用いたプログラムを説明する図

25

である。

【図１３】全体ブロック図である。

【図１４】整数演算ユニットのブロック図である。

【図１５】浮動小数点演算ユニットのブロック図である。

【図１６】浮動小数点レジスタファイルのブロック図である。

【図１７】浮動小数点レジスタのブロック図である。

【図１８】浮動小数点レジスタ１ビット分の回路図である。

【図１９】浮動小数点レジスタのブロック図である。

【図２０】シャドウレジスタの動作を説明する図である。

【図２１】シャドウレジスタの動作を説明する図である。

【図２２】シャドウレジスタの動作を説明する図である。

【図２３】モード制御回路の動作を説明する図である。

【図２４】レジスタ読み出し制御回路の動作を説明する図である。

【図２５】競合検出回路のブロック図である。

【図２６】コンパイラの処理フローである。

【図２７】モード制御回路の詳細を示した図である。

26

【図２８】データキャッシュの詳細を示したものである。

【図２９】ロードストア演算でキャッシュミスを生じたときのパイプラインを示す図である。

【図３０】その他の実施例を示す図である。

【図３１】その他の実施例を示す図である。

【図３２】その他の実施例を示す図である。

【図３３】その他の実施例を示す図である。

【図３４】その他の実施例を示す図である。

【図３５】その他の実施例を示す図である。

【図３６】その他の実施例を示す図である。

【図３７】その他の実施例を示す図である。

【図３８】その他の実施例を示す図である。

【図３９】その他の実施例を示す図である。

【図４０】その他の実施例を示す図である。

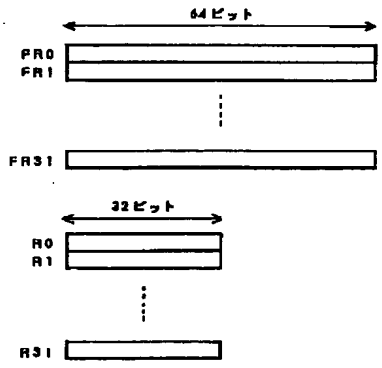
【図４１】図１を詳細化した図である。

【符号の説明】

１００…モード制御部、１０１…第１命令レジスタ、１０２…第２命令レジスタ、１０３…第３命令レジスタ、１０４…第４命令レジスタ、１０５…レジスタ読み出し制御、１０６…レジスタ書き込み制御、１０７…ファンクション制御、１０８…パイプライン制御、１０９…競合検出部。

【図2】

図 2

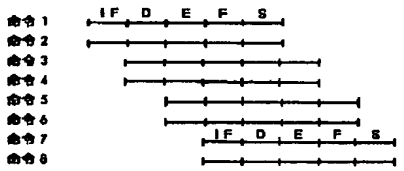


【図6】

図 6

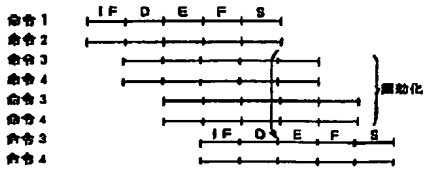
【図7】

図 7



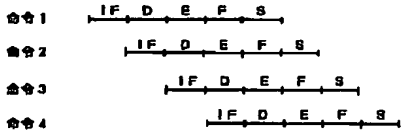
【図8】

図 8



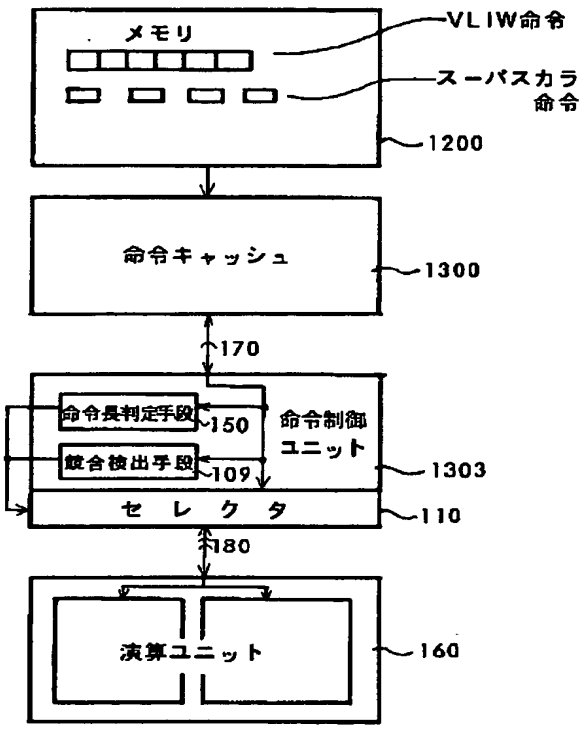
【図9】

図 9



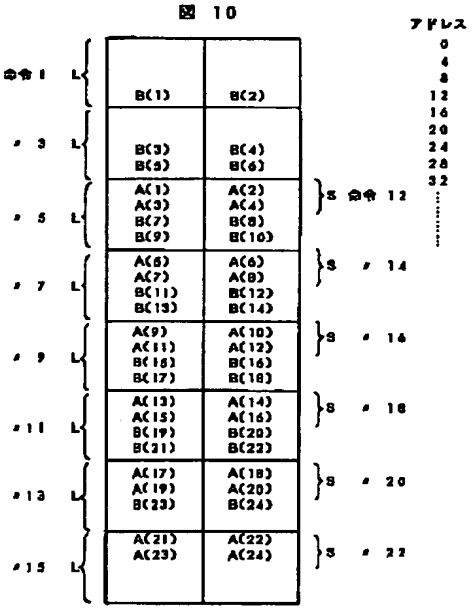
【図1】

図　1



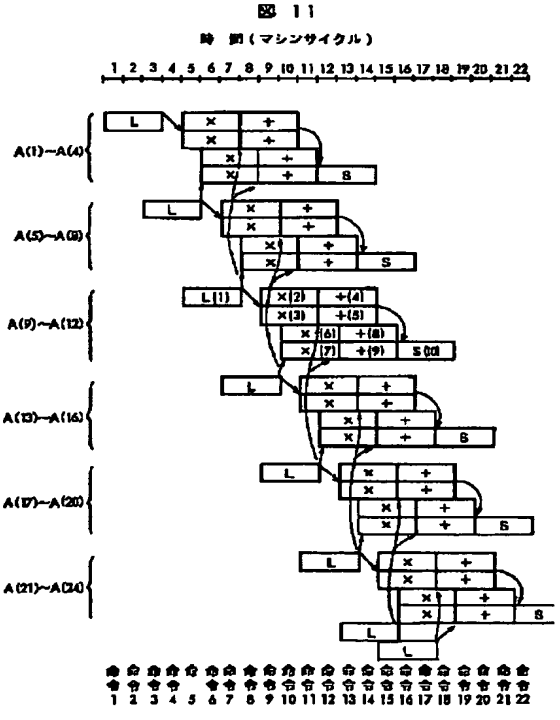
【図10】

図 10



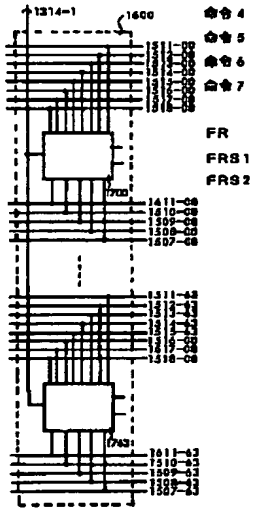
【図11】

図 11



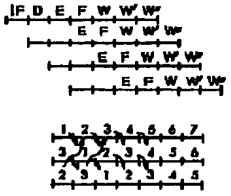
【図17】

図 17



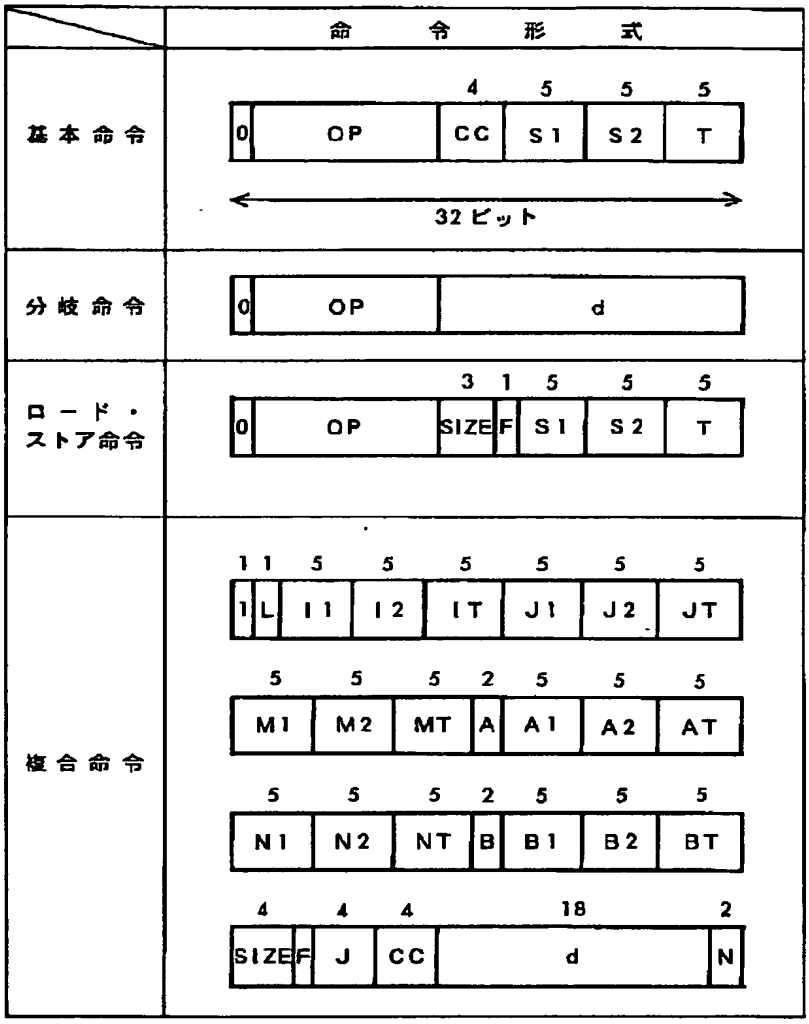
【図22】

図 22



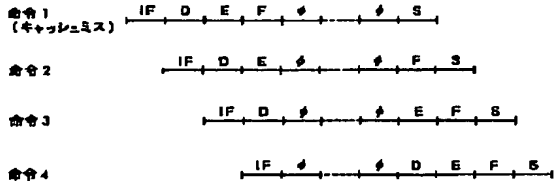
【図3】

図　3


命　令　形　式
基本命令
4 5 5 5
0 OP CC S1 S2 T
32ビット
分岐命令
0 OP d
ロード・ストア命令
3 1 5 5 5
0 OP SIZE F S1 S2 T
複合命令
1 1 5 5 5 5 5 5
1 L I1 I2 IT J1 J2 JT
5 5 5 2 5 5 5
M1 M2 MT A A1 A2 AT
5 5 5 2 5 5 5
N1 N2 NT B B1 B2 BT
4 4 4 18 2
SIZE F J CC d N

【図29】

図 29


命令1
(キャッシュミス)
IF D E F φ φ S
命令2
IF D E φ φ F S
命令3
IF D φ φ E F S
命令4
IF φ φ D E F S

【図4】

図　4

	ニモニック	動　作	補　足
基本命令	ADD　S1,S2,T	R(S1)+R(S2)→R(T)	CCフィールド
	SUB　S1,S2,T	R(S1)−R(S2)→R(T)	
	AND　S1,S2,T	R(S1)∪R(S2)→R(T)	
	OR　S1,S2,T	R(S1)∩R(S2)→R(T)	
	EOR　S1,S2,T	R(S1)(+)R(S2)→R(T)	
	NOT　S1,T	R(S1)→R(T)	
	FADD　S1,S2,T	FR(S1)+FR(S2)→FR(T)	
	FSUB　S1,S2,T	FR(S1)−FR(S2)→FR(T)	
	FMUL　S1,S2,T	FR(S1)×FR(S2)→FR(T)	
	NOP	何もしない。	
分岐	BRA　d	PC+d→PC	
ロードストア命令	ST　S1,S2	R(S1)をR(S2)でさされるメモリに書く	SIZEフィールド
	LD　S1,S2,T	R(S1)+R(S2)でさされるメモリ上のデータをR(T)に書く	
	FST　S1,S2	FR(S1)をR(S2)でさされるメモリに書く	
	FLD　S1,S2,T	R(S1)+R(S2)でさされるメモリ上のデータをFR(T)に書く	

コード	意　味
0000	常に不成立
0001	常に成立
0010	＝
0011	結果が正
0100	結果が負
⋮	⋮

コード	意　味
000	1ワード
001	2ワード
010	4ワード
011	8ワード
100	16ワード

注）∩…ビット毎のAND　∪…ビット毎のOR　(+)…ビット毎の排他的OR
PC…プログラムカウンタ　R(×)…レジスタ番号×の汎用レジスタ
FR(×)…レジスタ番号×の浮動小数点レジスタ

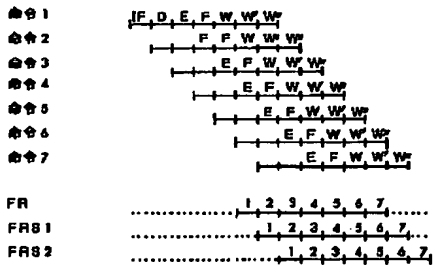
【図5】

図　5

	動　　作	補　　足
ロード・ストア演算	I1,I2,SIZE,F,ITフィールドを基本命令のS1,S2,SIZ,F,Tフィールドをみなして、ロード,ストアを実行	L＝1…ロード,L＝0…ストア F＝1…FR,F＝0…R
整数演算	J≠"1111"の時 R(J1)とR(J2)にJフィールドで示される演算を施して、R(JT)に格納	Jフィールド 0000 ADD,0001SUB 0010 AND,
	J≠"1111"の時 R(J1)+R(J2)でささされるデータが、キャッシュに無ければ、メモリからキャッシュに転送	
第1浮動小数点演算	FR(M1)×FR(M2)→FR(MT)	
第2浮動小数点演算	FR(A1)とFR(A2)にAフィールドで示される演算を施してFR(AT)に格納	Aフィールド コード：意味 00：FADD 01：FSUB
第3浮動小数点演算	FR(N1)×FR(N2)→FR(NT)	
第4浮動小数点演算	FR(B1)とFR(B2)にBフィールドで示される演算を施してFR(BT)に格納	BフィールドはAフィールドに同じ
フロー制御	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	Nフィールド コード：意味 00：0：NOP挿入 01：1：〃 10：2：〃 11：3：〃

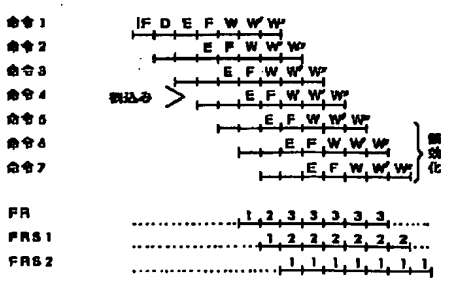
【図20】

図 20



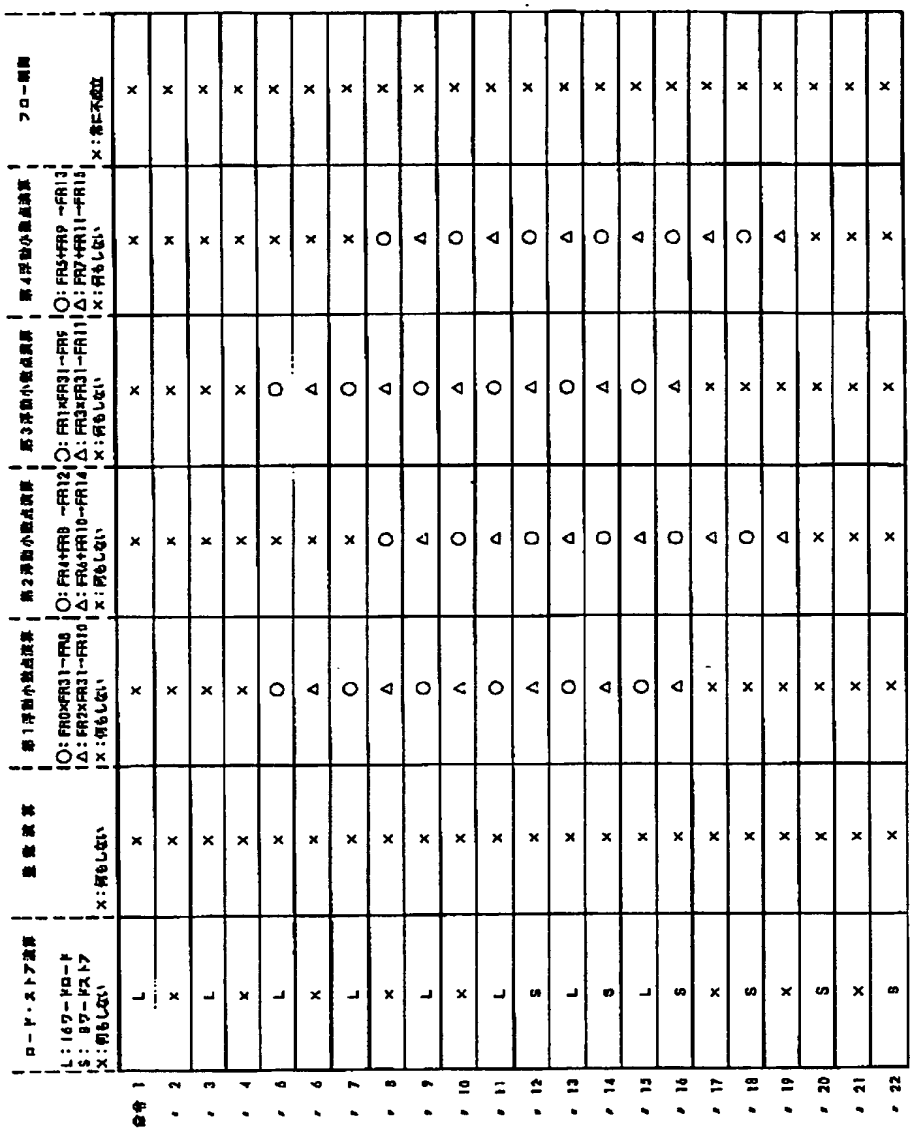
【図21】

図 21



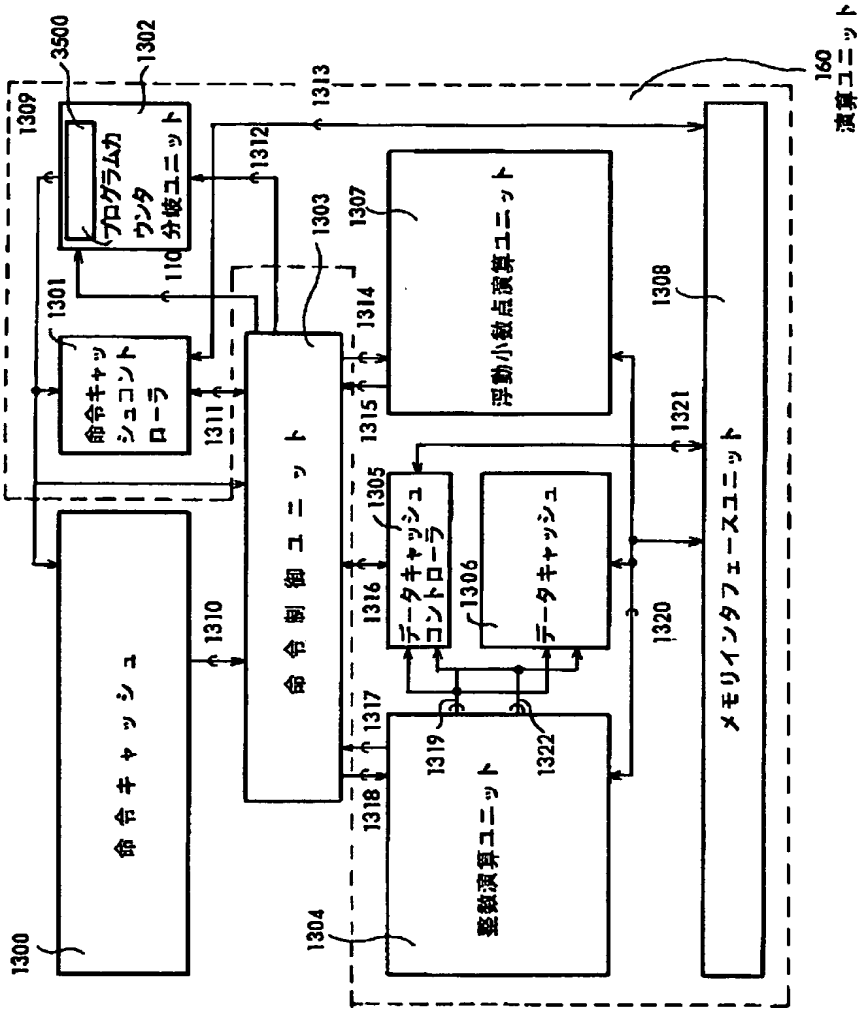
【図12】

図　12


ロード・ストア選択
L：ロード
S：ストア
×：何もしない
基底演算
×：何もしない
第1浮動小数点演算
○：FR0×FR31－FR8
△：FR2×FR31－FR10
×：何もしない
第2浮動小数点演算
○：FR4×FR8 －FR12
△：FR6×FR10－FR14
×：何もしない
第3浮動小数点演算
○：FR1×FR31－FR5
△：FR3×FR31－FR11
×：何もしない
第4浮動小数点演算
○：FR5×FR9 －FR13
△：FR7×FR11－FR15
×：何もしない
フロー制御
×：未正不定
命令 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

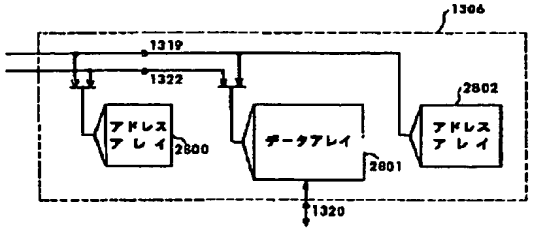
【図13】

図 13



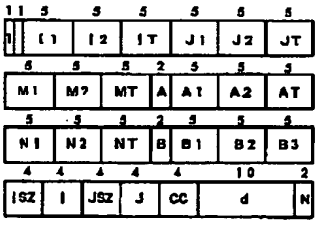
【図28】

図 28



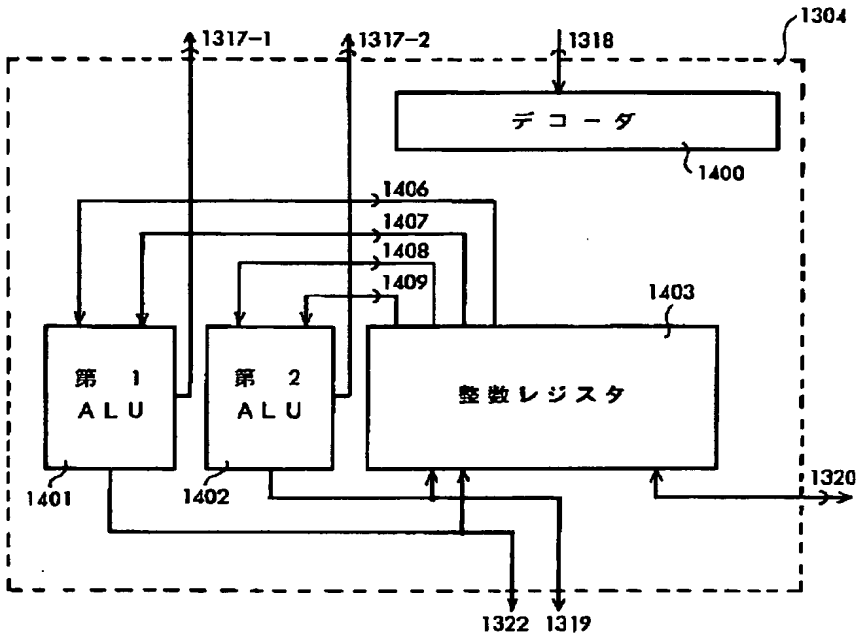
【図36】

図 36



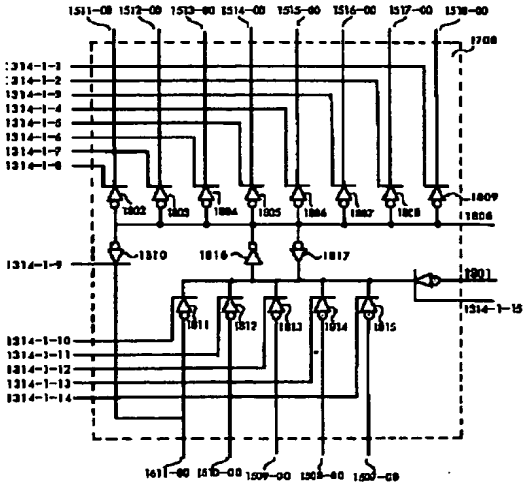
【図１４】

図 14



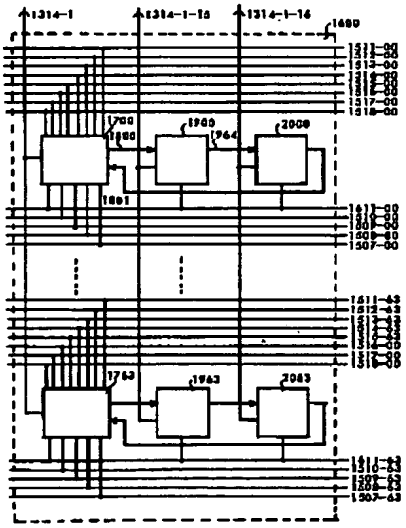
【図１８】

図 18



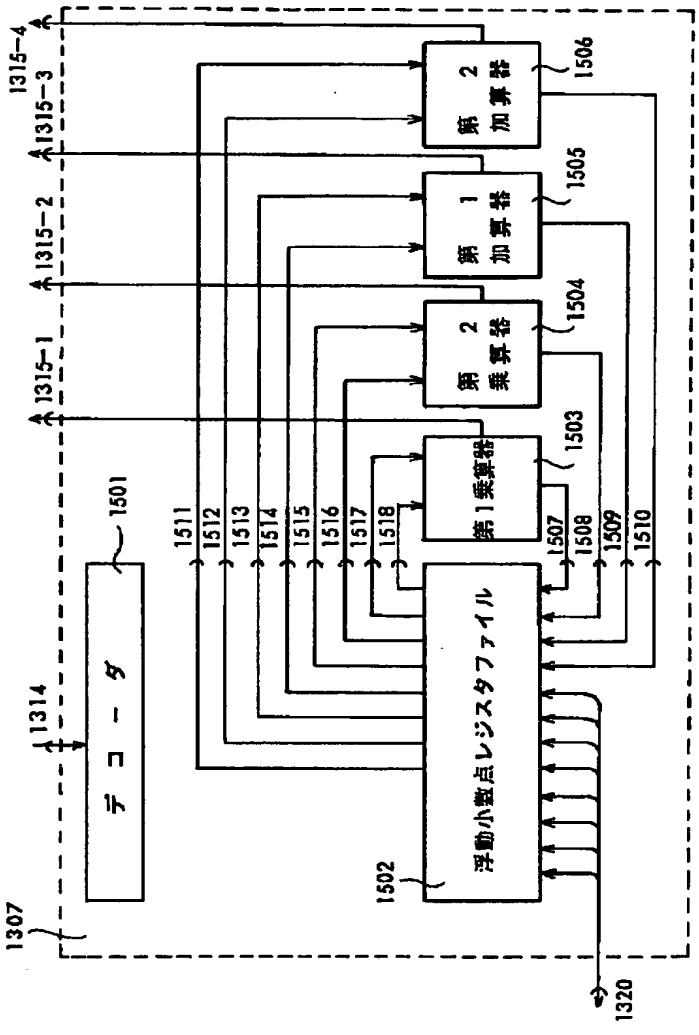
【図１９】

図 19



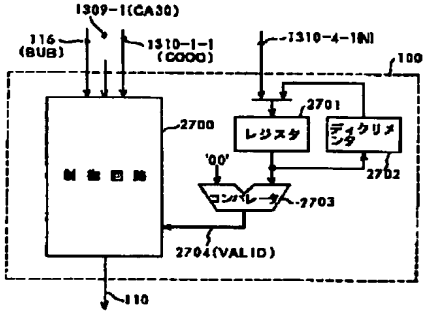
【図１５】

図 15



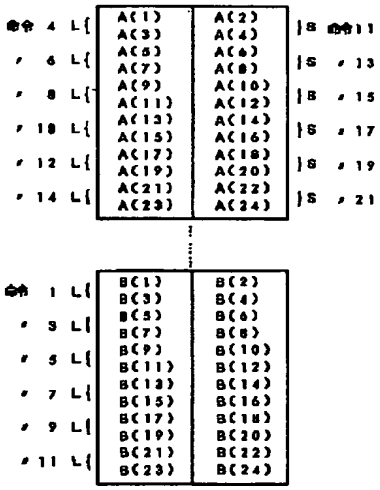
【図２７】

図 27



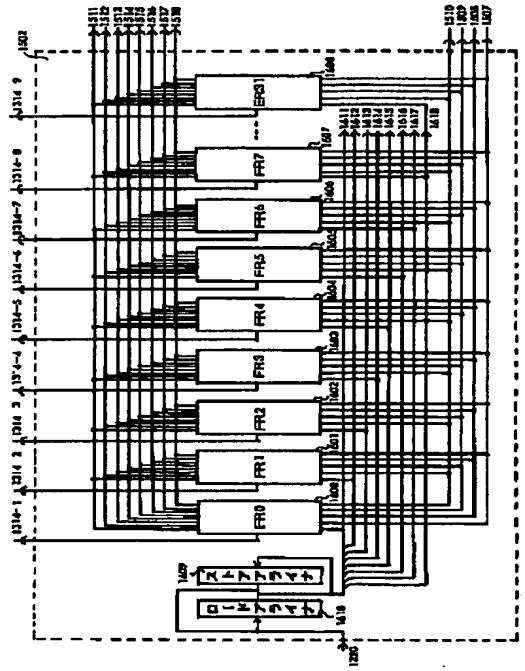
【図３８】

図 38



【図16】

図 16



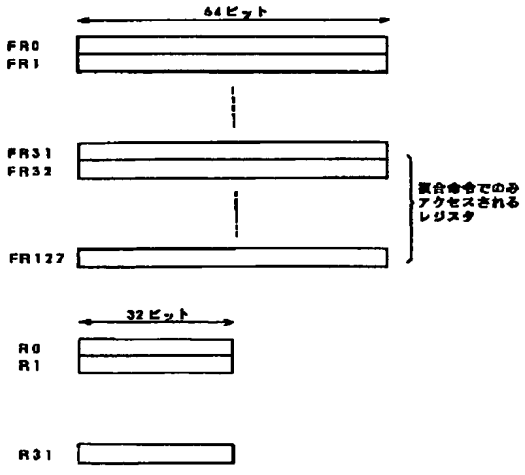
【図23】

図 23

モード	入力 BUB (116)	入力 C000 (1310-1)	入力 CA30 (1309-1)	出力 第1命令レジスタ	出力 第2命令レジスタ	出力 第3命令レジスタ	出力 第4命令レジスタ	出力 PC
1語命令モード1	0	0	0	C0	C1	−	−	+2
1語命令モード2	0	0	1	C2	C3	−	−	+2
4語命令モード	0	1	*	C0	C1	C2	C3	+4
競合モード	1	*	*	前サイクルと同じ	前サイクルと同じ	前サイクルと同じ	前サイクルと同じ	+0
無効命令モード	*	*	*	NOP	NOP	NOP	NOP	+0

【図30】

図 30



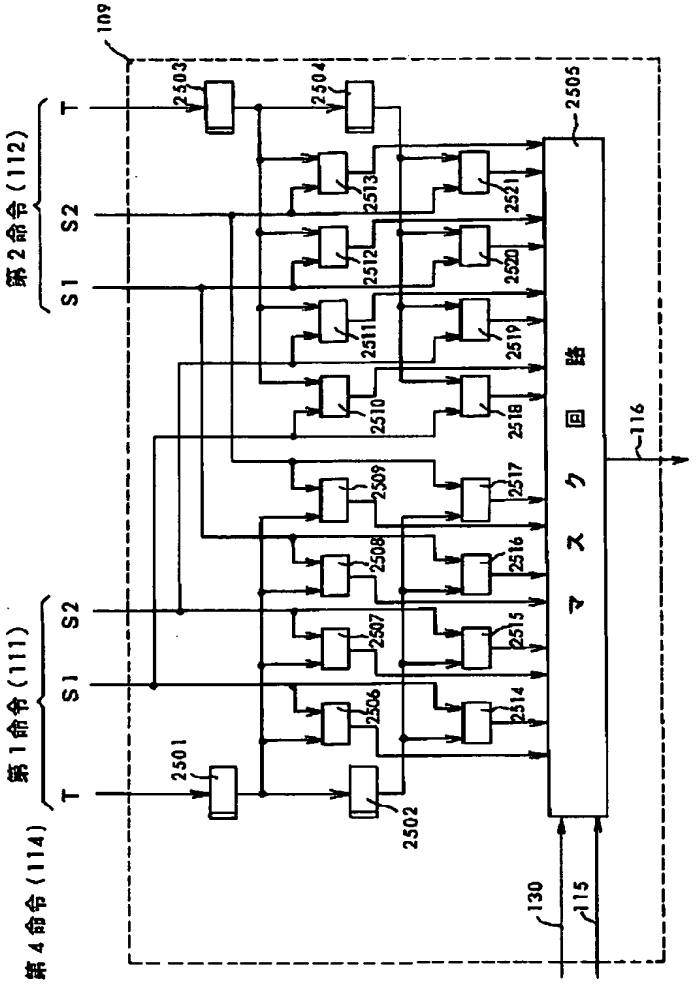
【図24】

図 24

演算器	ソース	1語長命令時	4語長命令時
第1ALU	1	J 1	I 1
	2	J 2	I 2
第2ALU	1	A 1	J 1
	2	A 2	J 2
第1乗算器	1	J 1	M 1
	2	J 2	M 2
第2乗算器	1	A 1	N 1
	2	A 2	N 2
第1加算器	1	J 1	A 1
	2	J 2	A 2
第2加算器	1	A 1	B 1
	2	A 2	B 2

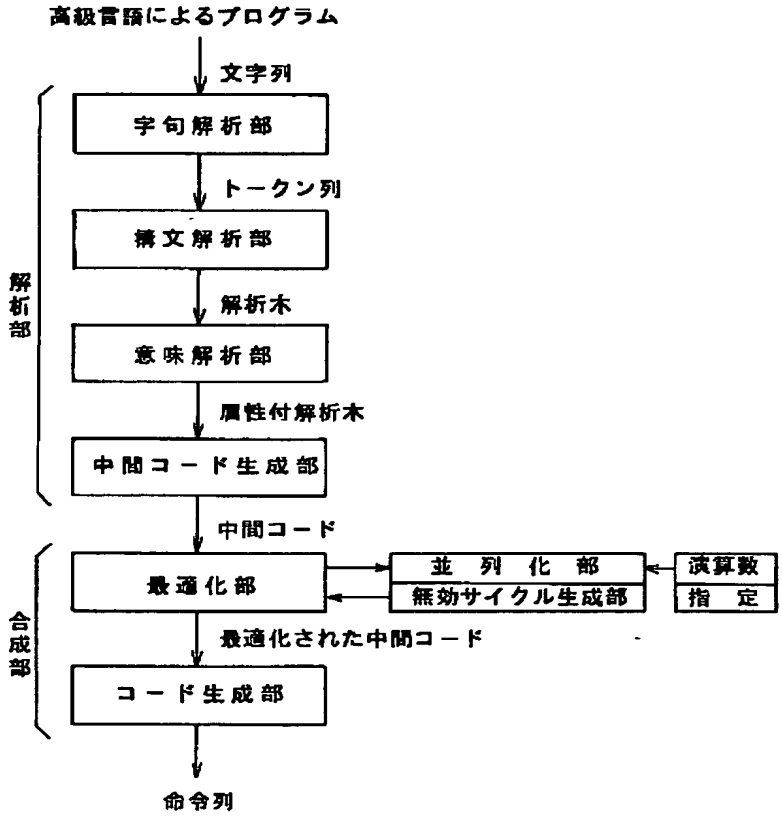
【図25】

図 25


109
第2命令(112)
T
S2
S1
第1命令(111)
S1
S2
T
第4命令(114)
2501
2502
2503
2504
2505
マスク回路
116
130
115
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521

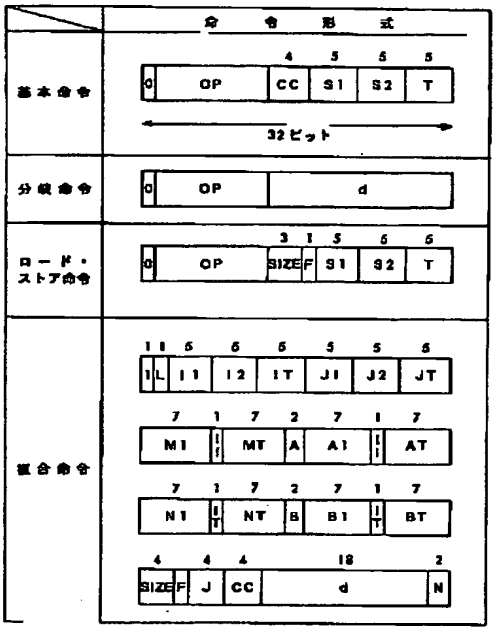
【図26】

図 26


高級言語によるプログラム
文字列
字句解析部
トークン列
構文解析部
解析木
意味解析部
属性付解析木
中間コード生成部
解析部
中間コード
最適化部
並列化部
無効サイクル生成部
演算数指定
最適化された中間コード
コード生成部
合成部
命令列

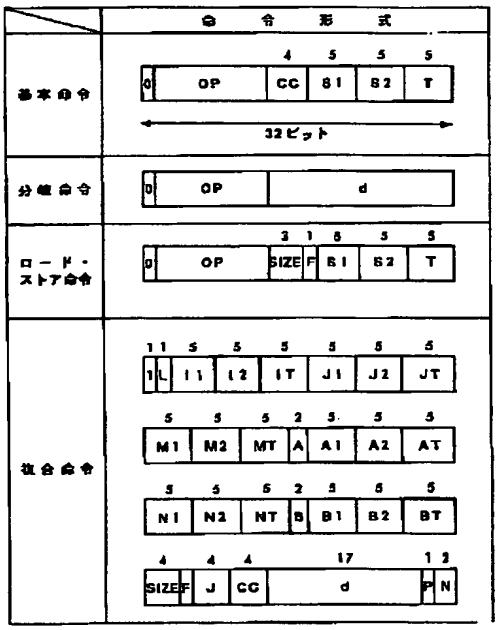
【図31】

図 31



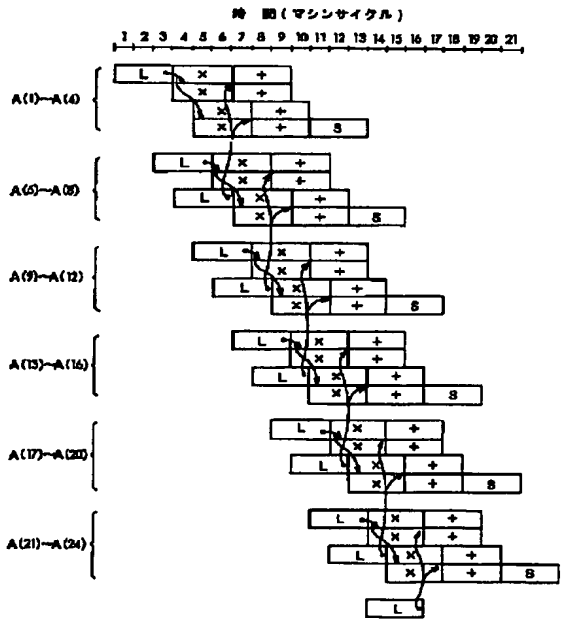
【図33】

図 33



【図39】

図 39



【図３２】

図　３２

<table>
<tr><th></th><th>動　　作</th><th>補　　足</th></tr>
<tr><td>ロード・ストア演算</td><td>I1，I2，SIZE，F，ITフィールドを基本命令のS1，S2，SIZ，F，Tフィールドをみなして，ロード，ストアを実行</td><td>L＝1…ロード，L＝0…ストア，F＝1…FR，F＝0…R</td></tr>
<tr><td rowspan="2">整数演算</td><td>J≠'1111'の時
R(J1)とR(J2)にJフィールドで示される演算を施して，R(JT)に格納</td><td>Jフィールド
0000 ADD，0001 SUB
0010 AND，</td></tr>
<tr><td>J≠'1111'の時
R(J1)+R(J2)でさされるデータが，キャッシュに無ければ，メモリからキャッシュに転送</td><td>JTフィールド
転送するデータの語数を示す。</td></tr>
<tr><td>第1浮動小数点演算</td><td>FR(M1)×FR(MT)→FR(MT)</td><td></td></tr>
<tr><td>第2浮動小数点演算</td><td>FR(A1)とFR(AT)にAフィールドで示される演算を施してFR(AT)に格納</td><td>Aフィールド
<table><tr><th>コード</th><th>意　味</th></tr><tr><td>00</td><td>FADD</td></tr><tr><td>01</td><td>FSUB</td></tr></table></td></tr>
<tr><td>第3浮動小数点演算</td><td>NR(N1)×FR(NT)→FR(NT)</td><td></td></tr>
<tr><td>第4浮動小数点演算</td><td>FR(B1)とFR(BT)に3フィールドで示される演算を施してFR(BT)に格納</td><td>BフィールドはAフィールドに同じ</td></tr>
<tr><td>フロー制御</td><td>整数演算命令の結果をCCフィールドに従い判定し，条件成立ならPC＋dに分岐</td><td>Nフィールド
<table><tr><th>コード</th><th>意　味</th></tr><tr><td>00</td><td>0：NOP挿入</td></tr><tr><td>01</td><td>1：〃</td></tr><tr><td>10</td><td>2：〃</td></tr><tr><td>11</td><td>3：〃</td></tr></table></td></tr>
</table>

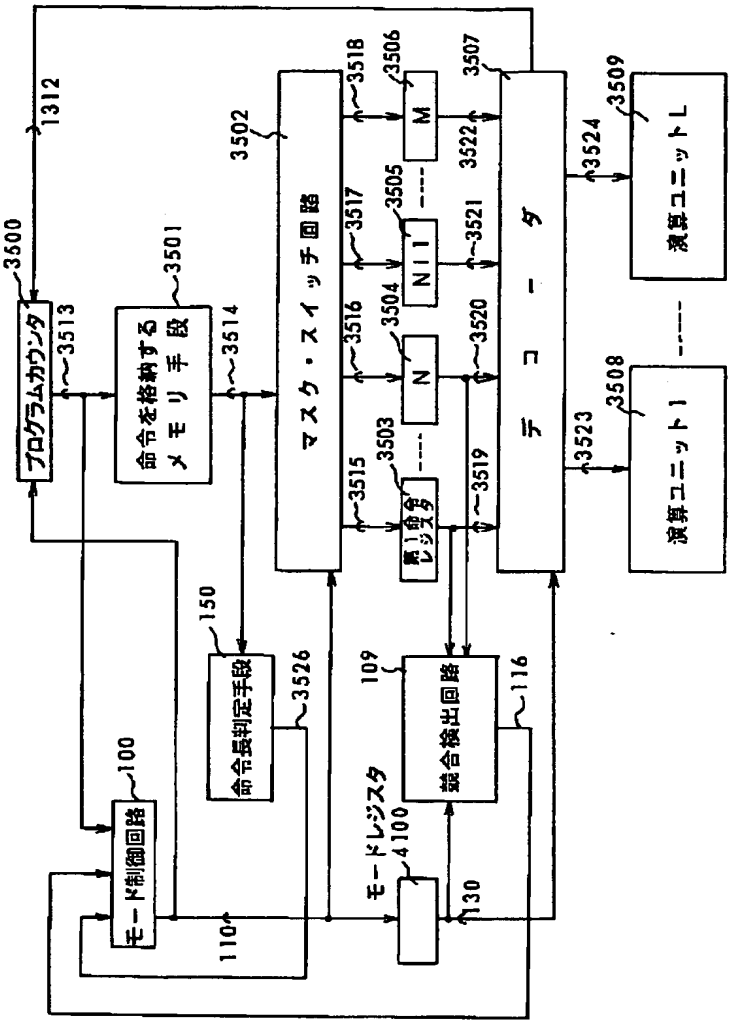
【図34】

図 34

	動　　作	補　　足
ロード・ストア演算	I1,I2,SIZE,F,ITフィールドを基本命令のS1,S2,SIZ,F,Tフィールドをみなして、ロード、ストアを実行	L＝1…ロード，L＝0…ストア F＝1…FR，F＝0…R P＝1…次のラインをプリフェッチ
整数演算	J≠"1111"の時 R(J1)とR(J2)にJフィールドで示される演算を施して、R(JT)に格納	Jフィールド 0000 ADD，0001SUB 0010 AND，
第1浮動小数点演算	FR(M1)×FR(M2)→FR(MT)	
第2浮動小数点演算	FR(A1)とFR(A2)にAフィールドで示される演算を施してFR(AT)に格納	Aフィールド コード：意味 00：FADD 01：FSUB
第3浮動小数点演算	FR(N1)×FR(N2)→FR(NT)	
第4浮動小数点演算	FR(B1)とFR(B2)にBフィールドで示される演算を施してFR(BT)に格納	Bフィールドは Aフィールドに同じ
フロー制御	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	Nフィールド コード：意味 00：0:NOP挿入 01：1:〃 10：2:〃 11：3:〃

【図35】

図　35


プログラムカウンタ
3513
命令を格納するメモリ手段
3501
3514
マスク・スイッチ回路
3502
3516
3517
3518
3504
3505
3506
N
N11
M
3520
3521
3522
デコーダ
3507
3524
演算ユニットL
3509
演算ユニット1
3508
3523
3519
3515
3503
第1命令レジスタ
1312
3500
150
命令長判定手段
3526
109
競合検出回路
116
100
モード制御回路
モードレジスタ
4100
130
110

【図37】

図 37

フィールド	動　作	補　足
第1整数演算 I1,I2,IT,ISZ,I	R(I1)とR(I2)を演算して R(IT)に格納	
第2整数演算 J1,J2,JT,JSZ,J	R(J1)とR(J2)を演算して R(JT)に格納	
第1浮動小数点演算 M1,M2,MT	FR(M1)×FR(M2)→FR(MT)	
第2浮動小数点演算 A1,A2,AT,A	FR(A1)とFR(A2)を演算して FR(AT)に格納	
第3浮動小数点演算 N1,N2,NT	FR(N1)×FR(N2)→FR(NT)	
第4浮動小数点演算 B1,B2,BT,B	FR(B1)とFR(B2)を演算して FR(BT)に格納	
フロー制御 CC,d	整数演算命令の結果をCCフィールドに従い判定し、条件成立ならPC+dに分岐	
NOP制御 N	N個NOPを挿入	

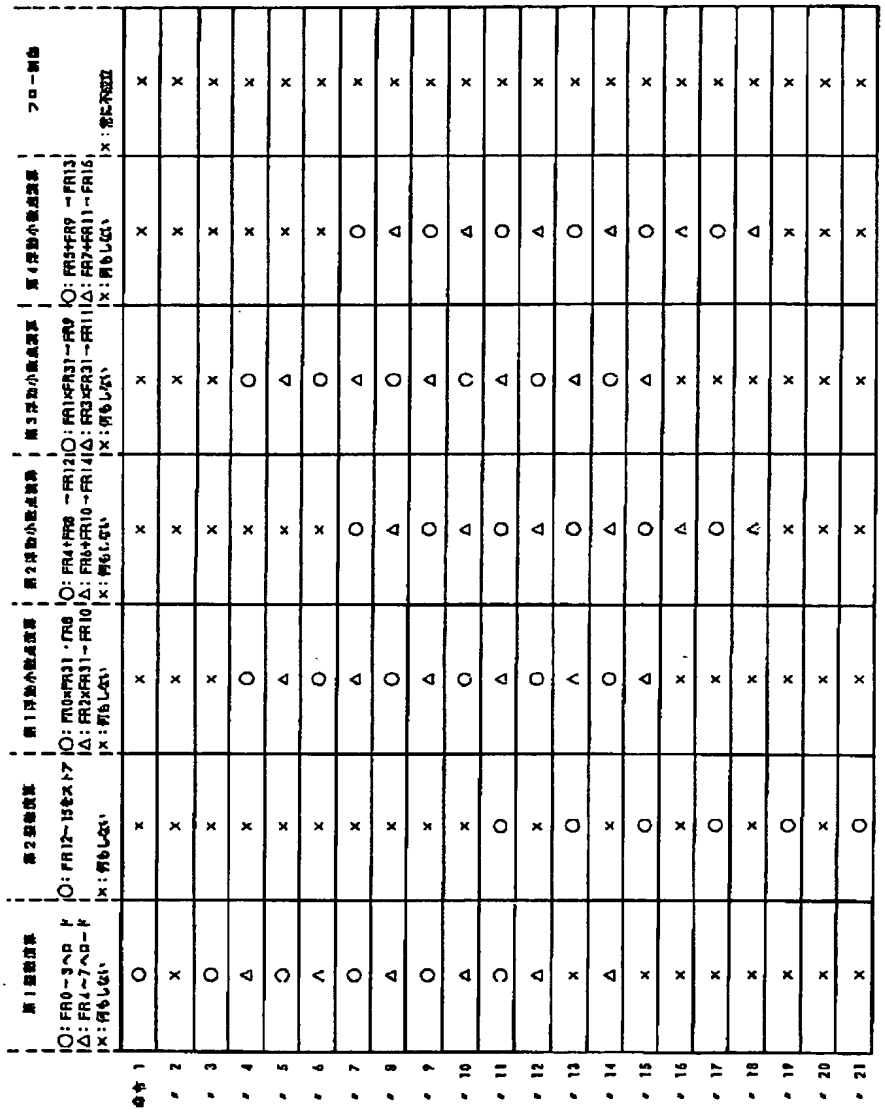
I,J	意　味
0000	ロード
0001	ストア
0010	プリフェッチ
0011	ADD
0100	SUB
⋮	⋮

ISZ,JSZ	意　味	
0000	1語	整数
⋮	⋮	整数
0111	256語	整数
1000	1語	浮動小数点数
⋮	⋮	浮動小数点数
1111	256語	浮動小数点数

A,B	意　味
00	FADD
01	FSUB

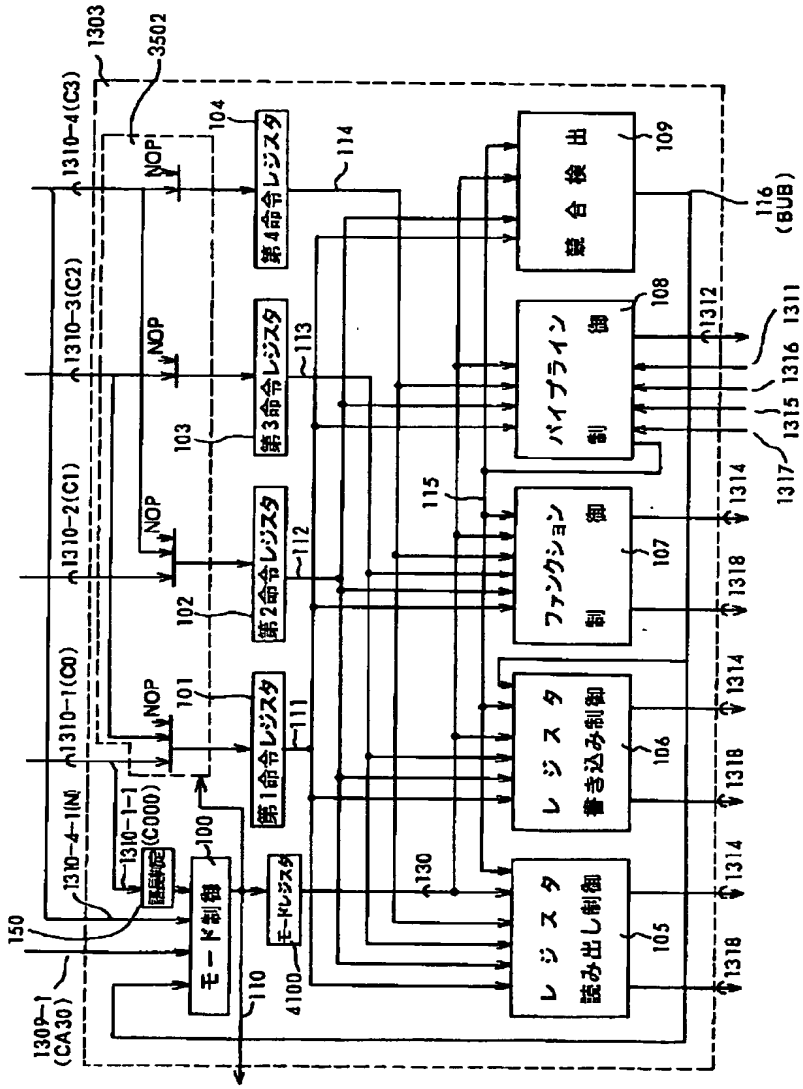
【図40】

図 40


第1浮動小数点演算
第2浮動小数点演算
第3浮動小数点演算
第4浮動小数点演算
フロー制御
○：FR0～3のどれか
△：FR4～7のどれか
×：何もしない
○：FR12～15のどれか
×：何もしない
○：FR0=FR31・FR8
△：FR2×FR31→FR10
×：何もしない
○：FR4+FR8 →FR12
△：FR6+FR10→FR14
×：何もしない
○：FR1×FR31→FR9
△：FR3×FR31→FR11
×：何もしない
○：FR5+FR9 →FR13
△：FR7+FR11→FR15
×：何もしない
×
命令 1
〃 2
〃 3
〃 4
〃 5
〃 6
〃 7
〃 8
〃 9
〃 10
〃 11
〃 12
〃 13
〃 14
〃 15
〃 16
〃 17
〃 18
〃 19
〃 20
〃 21

【図４１】

図　４１



フロントページの続き

(72)発明者　山田　弘道
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内

(72)発明者　前島　英雄
茨城県日立市久慈町4026番地　株式会社日立製作所日立研究所内